Eureka 3

Manual

Acknowledgements, Credits and Copyright

Eureka 3 is published by
Longman Logotron
124 Cambridge Science Park
Milton Road
Cambridge CB4 4ZS
Telephone (01223) 425558

Documentation: Sue Jansons, Doug Harrison, David Smith and Smith Owen Associates

Programming: Doug Harrison

Eureka

ISBN 0582 273 102

Please read carefully before using Eureka:

Logotron Limited provides the program Eureka under the rules of copyright, and licenses its use to the purchaser. This licence means that, unless you have Logotron's permission to the contrary:

(a) you may use the program on a single computer, or move the program and use it on another computer, but you may not use the program on more than one computer at a time;

(b) you may make one copy only of the program for backup purposes;

(c) you may not make any copy of the documentation.

Disclaimer

No warranty is made with respect to the program, its quality or performance, its merchantability or fitness for any particular purpose. The program is sold 'as is'. You, the user, assume responsibility for the selection of the program to achieve your intended results, and for the installation, use of and results obtained from the program.

What's new in Eureka 3?

New features

- Dual toolbar provides more facilities at the touch of a button
- Import and export of files from and to Microsoft® Excel on computers running Microsoft® Windows™ or Apple® Macintosh® computers
- Export of worksheets into Impression™ Style, with OLE (Object Linking and Embedding) facility
- Charts can be embedded into worksheets, or created in separate windows as before
- Graphics can be embedded into worksheets
- Header and footer margins can be different; headers and footers are now easier to enter
- Cell values can be recalculated automatically before saving or printing
- Iteration feature allows circular references between cells
- A new Utilities menu, providing a natural grouping of some new and existing functions.

All these new features are fully described in the following pages.

Enhancements

- Faster moving of cells
- Faster redrawing of charts
- Background printing, allowing you to continue working while your worksheet or chart is printed.

The toolbars

Eureka now has two toolbars. One appears when you run the application:

saves the worksheet with the current filename. If you have not saved the worksheet before, a Save as dialogue box appears so that you can give it a name.

displays the Print dialogue box

is equivalent to choosing Edit ➪ Cut

is equivalent to choosing Edit ➪ Copy

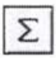
is equivalent to choosing Edit ➪ Paste

displays the Fill dialogue box

removes or restores the display of row and column headings

removes or restores the grid between cells

recalculates cell values (you only need to use this if automatic recalculation is not set)



inserts the SUM formula into the data entry area

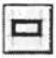
allows you to drag out an area in which to display the currently selected cells as a chart. When you have done this, the New chart dialogue box appears.

At the right-hand end of the toolbar is a button that switches between the two toolbars. Click on this to show the second toolbar, which contains buttons for changing the way your data is displayed: its style, font, alignment, cell borders and fill.

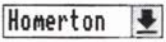
allows you to choose a number format for the selected cells


aligns the cell contents to left, centre or right


changes the background of the cell(s) to a light grey

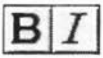

changes the border of the cell(s) to a fine line


changes the lower border of the cell to a fine line. If several cells are selected, each one is given a border.



allows you to choose a font for the selected cells

allows you to choose a point size for the content of the selected cells



changes the content of the selected cells to bold or italic style.

Exporting Eureka worksheets to Microsoft Excel

To export a Eureka worksheet to Excel:

1. Choose Save as from the File menu.
2. In the File type scroll box, choose Excel 3 or Excel 4, according to the version of Excel you will be using. (Choose either if you are using Excel 5.)
3. Insert an MS-DOS format disc into your floppy disc drive, and open a directory display for it. If you are using RISC OS 2, you will need to run DOSFS, or a similar program, to enable your computer to read and write DOS format discs.
4. Drag the icon from the Save as dialogue box to the directory display.

You can then insert the floppy disc into the drive of your MS-DOS PC and load the file into Excel.

To export only part of a worksheet, select the cells you want to export and tick the Limit to selection box in the Save as dialogue box.

Importing Excel worksheets into Eureka

To import an Excel 3 or 4 worksheet into Eureka, you must first ensure that its file type is correct. To do this:

1. Locate the file's icon in the directory display.
2. Click Menu with the pointer over the icon, select File 'Filename' ➪ Set type, and look at what appears in the Set type box. This is likely to be 'DOS' if you have just copied the file from a PC running Windows.
3. Delete the existing file type if necessary, type 'Excel', and press Enter.
4. Drag the file's icon onto the Eureka icon on the icon bar.

If you use Excel 5, you will only be able to import a single sheet in a workbook at a time. In addition, any attributes of your Excel 5 worksheet that were not available in Excel 4 will not be imported.

Importing and exporting from Macintosh computers

To import from Excel running on Macintosh computers, first use the Macintosh to transfer the file to an MS- DOS disc, then follow the steps given above for Windows. You may need to use a utility program such as Apple File Exchange on your Macintosh, to enable it to read and write DOS format discs.

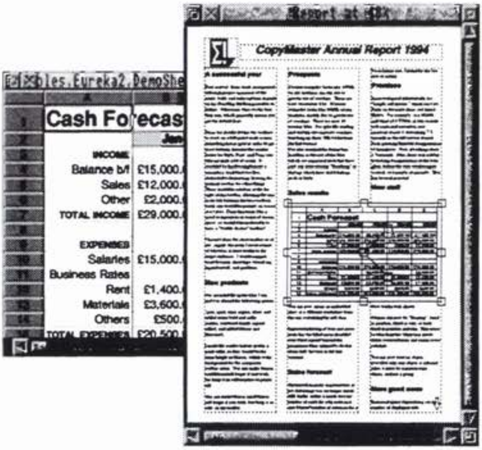
To export to Excel on a Macintosh, you must similarly transfer the file via an MS-DOS disc.

Exporting a worksheet into Impression Style

You can export a worksheet into Impression Style (a word processor and desktop publishing application). To do this:

1. Open the Impression Style document where you want your Eureka worksheet to appear.
2. Returning to Eureka, choose Save as from the File menu.
2. In the File type scroll box, choose Impression OLE.
3. Drag the icon from the Save as dialogue box onto the Impression Style document.

When you have loaded the file into an Impression Style document, you may want to edit it further. To do this, hold Ctrl and click on the worksheet within Impression Style. This opens the worksheet in Eureka. When you have finished editing, close Eureka and your Impression Style document will be updated with the edited version of your worksheet.



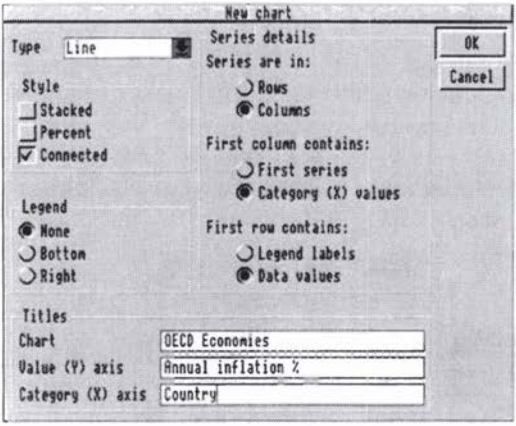
Charts in Eureka 3

In earlier versions of Eureka, charts were created in separate windows. You can still do this in Eureka 3, but you can also create a chart that becomes part of your worksheet: it is 'embedded' in the worksheet.

Creating an embedded chart

1. Select the cells you want to chart.

2. Click on the Chart button in the toolbar.
3. Back in the worksheet, drag out a rectangle where you want your chart to appear (you can always resize or reshape this later if necessary).
4. The New chart dialogue box will appear.



- In the Type drop-down list box, select the type of chart you want to use.
- The options available in the Style box below depend on the type of chart you choose.
    - If you choose Area, Bar or Column, you will be able to choose from Stacked or Percent in the Style box below (if you choose Area, Stacked will be ticked by default, but you can change this if you like).
    - If you choose Line, Stacked and Connected will be ticked by default, but again, you can change these settings.
    - If you choose Scatter, Connected is ticked by default.
    - Finally, if you choose Pie, the Style options are all greyed out.
- In the Legend box, choose whether you want the legend to appear at the Bottom of the chart, or to its Right; if you don't want a legend at all, choose None.

- The Series details box allows you to specify whether the series in your selection are in rows or columns, whether the first column contains some of the data to be charted or the category values, and whether the first row in your selection contains labels or data values. Eureka will make sensible choices to use as defaults, based on the kind of data in the relevant rows or columns, so you can probably leave these choices unchanged.
- The Titles box allows you to type in titles for the chart itself and the axes.

6. Click on OK to display the chart in the rectangle you have drawn, or on Cancel to go back to the worksheet without displaying the chart.

To edit the chart, hold Ctrl while clicking on it. The chart borders are now highlighted; pressing Menu with the pointer on the chart displays the Chart menu, which is described on pages A-40–A-48 of the Eureka manual.

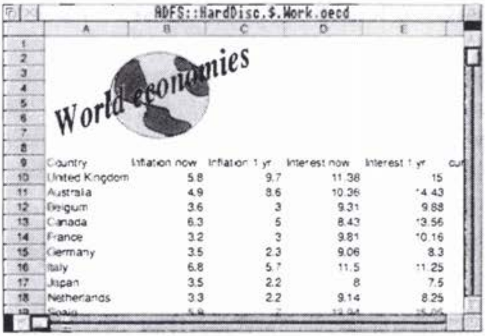
Creating a chart in a separate window

To create a chart in a separate window, as in earlier versions of Eureka, click Menu with the pointer anywhere on the tool bar or status bar, and select New document ⇨ Chart. This displays a bar chart using the default settings, which you can change if you wish using the Chart menu (see the Eureka manual and the chapter Creating Graphs and Charts for how to do this).

Embedding graphics into worksheets

In Eureka 3 you can enhance your worksheets with graphics. These can be !Draw files made up of lines, shapes and objects, sprites, or files produced by the ArtWorks application.

To embed a graphic, drag its icon from the directory viewer where it is saved, onto the worksheet where you want it to appear. Once it is in the worksheet, you can move, resize or reshape it by selecting it and dragging in the usual way, as in applications such as Draw.



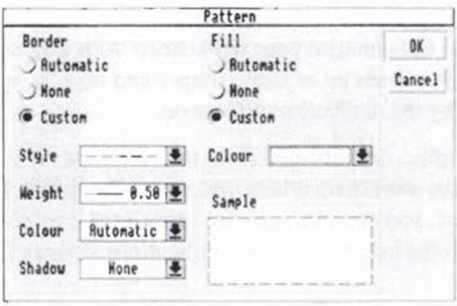
Using several copies of a graphic

If you want a graphic to appear several times in a worksheet, you do not need to import it more than once. Instead, import it once and make additional copies using Edit ➩ Copy; when you do this, Eureka stores only one copy of the graphic, however many times it appears. This will reduce the size of your Eureka file, especially if the graphic is large.

Changing the border and background of a graphic

Although you cannot edit the graphic itself once it is embedded in Eureka, you can change its border or background. To do this:

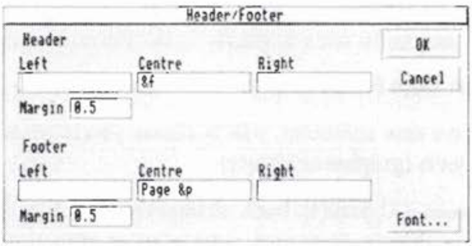
1. Select the graphic.
2. Press Menu and choose Format ➩ Background to display the Pattern dialogue box:



- Choose Automatic in the Border box to give the graphic a black border 0.5 pt wide; choose Automatic in the Fill box to give it a white background.
- Choose None in either box if you want the graphic to appear without a border or background.
- Choose Custom in the Border box to choose a style, weight (in points), colour and shadow style for the border. The choices you make (except for Shadow) are shown in the Sample box as you make them.
- Choose Custom in the Fill box to choose a background colour for the graphic.

Worksheet headers and footers

In earlier versions of Eureka, you could not alter the height of the header or footer for a printed worksheet. In Eureka 3, the Page setup dialogue box (displayed when you choose File ⇨ Page setup) contains Header and Footer buttons; press either of these to display the Header/Footer dialogue box. The initial position of the cursor when the dialogue box opens depends on which button you press, but you can set both the header and footer in the same dialogue box.



Enter the text you want to appear in the Left, Centre or Right aligned positions in the header or footer. The codes for inserting variables such as page number and date are given on page B-51 of the Eureka manual. You do not need to use the codes for positions (&1, &c, etc), as in Eureka 3 you type these components of the header or footer directly into the appropriate boxes.

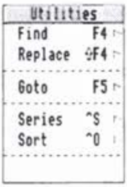
To set the font for the header and footer, press Font and choose the font, size and style in the usual way.

Set the height of the header or footer margin in the appropriate box. The units used will depend on the setting in the Eureka Settings file. To change this, hold Shift while double-clicking on the !Eureka application icon in the directory display. This opens the application as a directory; within this directory, double-click on the Resources sub-directory, in which you will see a file called Settings. Double-click on this to load it into a text editor.

The unit is set in the line reading Units = <unit>, where <unit> is the currently chosen unit. Edit this to your preferred unit and save and close the Settings file.

Utilities menu

Eureka 3 contains a new Utilities menu:



Find, Replace and Goto were formerly in the Formula menu.

Series and Sort were formerly in the Data menu.

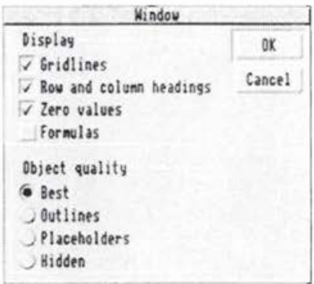
Object leads to a new submenu, which allows you to manipulate embedded objects (graphic or charts).

- Bring to front and Send to back change the stacking order of overlapping objects. However, none of these objects can be placed behind the worksheet data itself.
- Group links the selected objects so that they can be moved or resized as a single object.
- Ungroup unlinks the selected group so that its components can be manipulated independently.

- Placement leads to the Placement dialogue box, through which you can control changes to the size and position of the object when you insert or delete rows or columns. For example, if Floating object or Maintain size is selected when you insert a row above the object, the object will move down, staying within the group of cells where it was originally placed. If Floating object or Maintain position is selected when you insert a row in the middle of the object, the object will stretch. To fix the object at a specific size and position on the worksheet, make sure Maintain size and position is selected before you insert or delete cells.
- Save object leads to a normal Save as dialogue box, allowing you to export the object separately from the worksheet and other objects in it.

Window dialogue box

Display this dialogue box by choosing Options ⇨ Window.



The options in the Display box are described in the Eureka manual on pages A-33–A-34. The options in the new Object quality box allow you to choose how much detail you see in embedded objects:

- Best displays them in full.
- Outlines displays filled and patterned areas in outline only.
- Placeholders displays no details of the objects but shows their size and position.
- Hidden removes them from the display altogether (though you can still select them – if you know where they are).

Reducing the amount of detail displayed will reduce redrawing time, which can be significant if you have several large graphics.

Setting the width of a font

This is done, as before, with the Font dialogue box, displayed when you choose Format ➪ Font. The only difference is that you now state the width you want as a percentage of the font's height. For example, to obtain a font stretched widthways, enter a figure greater than 100 in the box. To obtain a compressed font, enter a figure less than 100.

Calculation

Automatic and manual recalculation

As in earlier versions of Eureka, you can choose whether cell values should be recalculated automatically every time you change a value. To do this, choose Options ➪ Calculation to display the Calculation dialogue box:

In the Mode box, you can choose between Automatic and Manual calculation, as before. If you choose Manual, you can trigger recalculation by clicking on the Recalculate button in the toolbar. However, when Manual is selected, you might inadvertently save or print your worksheet with inconsistent values. You can ensure that this does not happen by ticking Calculate before save in the Calculation dialogue box; when this is set, Eureka recalculates before saving or printing.

Iteration: calculating circular references

Eureka 3 also allows circular references between cells, so that a cell refers indirectly to itself. Often, such references will have been introduced by mistake, and when this is the case, you should of course correct them. However, in some circumstances you may want a circular reference. For example, suppose a salesperson's gross sales amount to £1,000. Her commission is calculated as 10% of net sales, which is defined as gross sales less net sales. The formulas you define in Eureka will look like this:

	A	B
1	Gross sales	1000
2	Net sales	B1–B3
3	Commission	B2*10%

B3's formula refers to B2, and vice versa. Eureka can make this calculation by carrying out repeated approximations. To enable this feature, open the Calculation dialogue box and make sure that Iteration is On. In addition, you can change the maximum number of iterations (100 is the largest number you should choose, in practice) and the tolerance value (the maximum change between iterations). The larger the maximum iterations, and the smaller the tolerance value, the longer the worksheet will take to recalculate.

Acknowledgements, Credits and Copyright

Eureka is published by:
Longman Logotron
124 Cambridge Science Park
Milton Road
Cambridge CB4 4ZS
Telephone (0223) 425558

Tutorial Documentation: Sue Jansons
Reference Documentation: Doug Harrison and David Smith
Programming: Doug Harrison

Eureka

ISBN 0582 097428

Please read carefully before using Eureka:

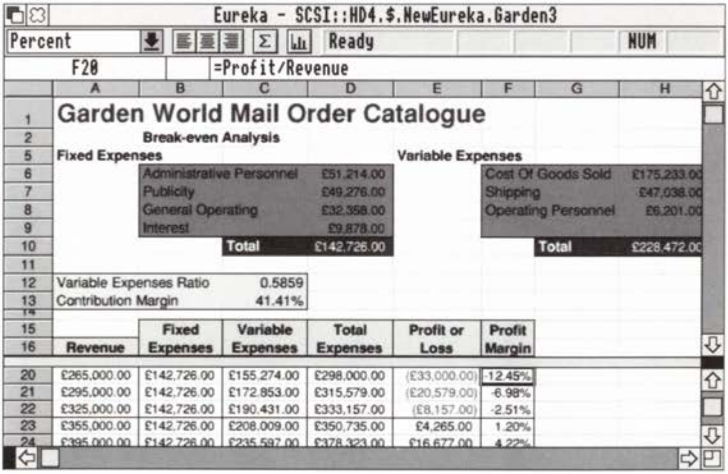
Contents

Eureka Reference Guide

Introduction

A spreadsheet is used to analyse and present numerical data, usually data of a financial nature.

A spreadsheet file is called a worksheet. The worksheet is organised into rows and columns forming a grid of cells. Each cell can contain words, numbers or a calculation. Like an accountant's sheet of paper, a blank worksheet can be formatted and labelled to create invoices, cash forecasts, conversion tables, and so on. An example spreadsheet, generated on Eureka, is shown below.



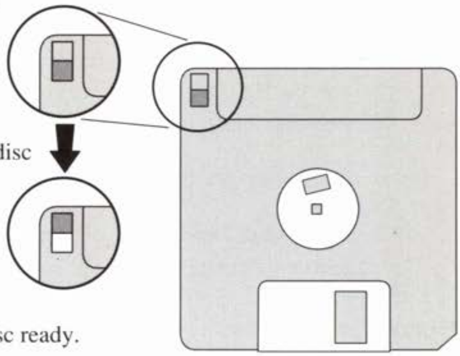
However, unlike a sheet of paper, a spreadsheet performs calculations automatically. When you insert a formula into a cell, it uses data stored in other cells to calculate and display the results. If you alter the data, the spreadsheet automatically updates the answers.

The speed with which a spreadsheet like Eureka can perform these calculations allows you to ask 'what if?'. By altering the data, you can model different circumstances and see instantly how they will affect the 'bottom line'; and because numbers are often clearer when presented graphically, Eureka supports a comprehensive range of charting facilities which rapidly turn your data into graphs and charts suitable for reports or presentations.

Getting Ready

Making a Backup copy of Eureka

It is strongly recommended that you make a working copy of the Eureka program disc.



• Write protect the original disc by sliding the small plastic tab to uncover the hole at the edge of the disc then insert it into the disc drive.

• Have a blank formatted disc ready.

• Position the cursor on the floppy disc icon and click the Menu button.

• Select the Backup option and follow the instructions on the screen.

Updating !System (RISC OS 2 users)

Like most RISC OS applications, when running under RISC OS 2, Eureka makes use of a special application called !System, which provides a number of important facilities. A copy of !System is supplied on the Eureka program disc.

Because !System has been updated from time to time, your existing copies may not be suitable for Eureka. If this is the case, when you try to load the program you will be given an error message.

The Eureka disc includes a special application, called !SysMerge, which allows you to update all your copies of !System to the new version.

• Insert the Eureka disc into the disc drive.

• Position the pointer on the disc drive icon. Click Select (the left mouse button) once.

• The Eureka directory viewer will appear. Double click on the SysMerge icon to open the !SysMerge window.

• Insert the disc containing the copy of !System you want to update, and click on the disc drive icon. (Hard disc users should open the directory viewer that contains their existing !System.)

• Drag the existing !System into the !SysMerge window.

• Following the instructions, drag the new !System (from the Eureka disc) into the !SysMerge window. Your existing !System will be updated. (You may be prompted to swap discs.)

• Close the !SysMerge application by clicking on its close icon.

If you have copies of !System on other discs, use the same procedure to update them all.

Hard Disc Users

To install Eureka on your hard disc:

• update your !System application.

• create a directory called 'Eureka' and copy the contents of the Eureka program disc.

For clarity, all instructions in this manual are given for a system with a single floppy disc drive.

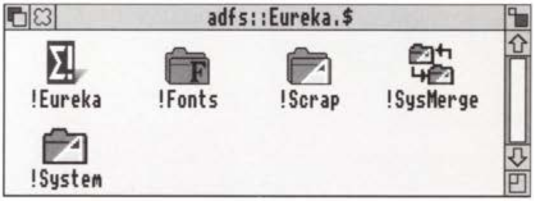
Getting Started

Loading Eureka

• Insert the Eureka program disc into the disc drive.

• Position the pointer on the disc drive icon.

• Click Select (the left mouse button) once. The Eureka directory viewer will be displayed on the screen.



• Double click on the Eureka icon. The Eureka icon will be installed on the icon bar.

• Single click on the Eureka icon. The Control Bar and a blank worksheet will appear on the screen.

(If an error message appears at this stage, see Updating !System, page 3.)

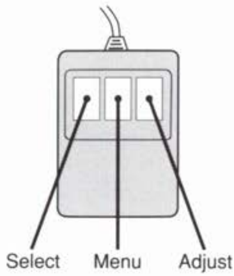
RISC OS

Eureka conforms to standard RISC OS procedures for loading and saving files, for moving, re-sizing, and scrolling windows, for opening menus and dialogues and selecting options from them.

This manual assumes that you are familiar with the RISC OS environment; if not, please consult the appropriate Acorn manual.

To take fullest advantage of all Eureka's facilities, you will require RISC OS 3 and should consider upgrading your computer if necessary.

The Mouse



Eureka makes extensive use of the mouse. This manual refers to the three mouse buttons as Select (left), Menu (middle) and Adjust (right).

The Cursor

As you use Eureka you will find that the cursor changes shape according to where it is and what it is doing.

The standard RISC OS **pointer** is used for selecting options from menus and responding to dialogues.

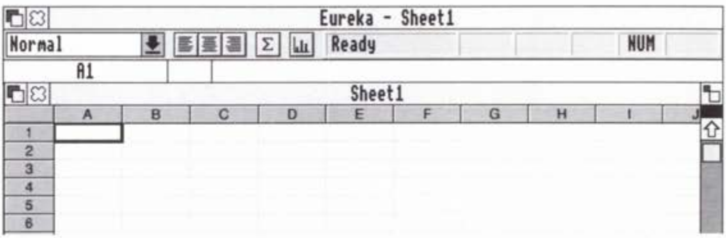
The **caret** appears whenever you are using a text field.

The **double headed arrow** appears when you use the pointer to re-size rows and columns or to split the worksheet window.

When the pointer is on the worksheet, it turns into the **worksheet cursor**. This is used to select, or 'highlight', cells.

In addition, the **cell cursor** indicates the current selected or 'active' cell.

The Worksheet



When you load Eureka, it displays a blank 'worksheet', labelled 'Sheet1'.

A worksheet is a spreadsheet file. It is divided into rows and columns forming a grid of **cells**. The rows are labelled with numbers (1, 2, 3...) and the columns with letters (A, B, C...).

The complete worksheet contains 16,384 rows and 256 columns (the first twenty-six columns are labelled A-Z, the next twenty-six AA-AZ, and so on). Only a small part of a sheet is displayed at any time, but you can scroll up and down and back and forth using the standard RISC OS scroll bars.

The Control Bar

The Eureka Control Bar runs across the top of the screen. It provides the following functions:

send to back

moves the control bar behind any other open windows.

close down Eureka

clicking on ✖ closes down the Eureka Control Bar and the current worksheet, but the Eureka icon remains on the icon bar. Clicking on the icon re-opens the Control Bar and opens a new worksheet.

title bar

displays name of current worksheet. If more than one worksheet is open, the title bar will show the name of the active sheet.

Tool/Status Bar

style menu

provides a number of basic options for formatting data - as currency or percentages, for example. Click on the arrow to display the complete range of built in styles available.

align left, centre, or right

used to realign data in the selected cell or cells.

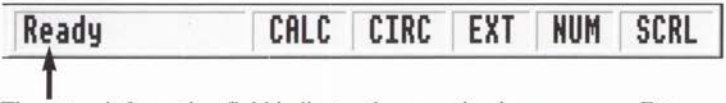
sum

click to insert SUM function into data entry bar.

chart/graph button

click to open chart window and display selected data in chart form.

Status Indicators



The status information field indicates the operation in progress — Enter, Edit, Copy, Point, etc. Otherwise, displays Ready.

CALC

in manual calculation mode, indicates that data has been altered but formulas have not yet been recalculated.

CIRC

indicates that a formula on the worksheet refers to its own address — a circular reference.

EXT

indicates extended mode (keyboard control) is selected.

NUM

indicates number lock on.

SCRL

indicates scroll lock on.

Data Entry Bar

displays the address of the selected cell (or the highlighted cell of a range)

click to abandon changes after editing the text field

click to confirm changes after editing the text field

data entry field

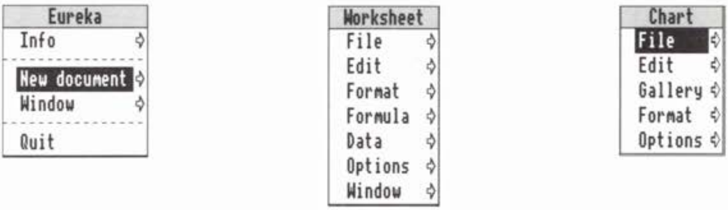
data appears here as you type it in. It is inserted into the selected cell when you press Return.

The Menus

Most of Eureka's functions are controlled via three menus. The Eureka Menu is opened by positioning the pointer on the Control Bar and clicking Menu (the middle mouse button).

The Worksheet Menu is opened by positioning the pointer on the worksheet and clicking Menu.

The Charts Menu controls Eureka's charting facilities, and is opened by positioning the pointer anywhere on a Chart window and clicking Menu.



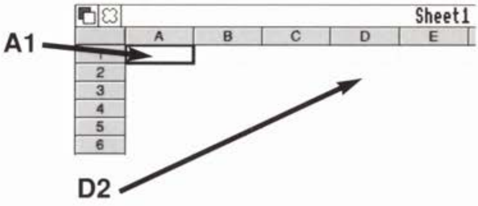
Related operations are grouped together on these menus and are controlled via submenus and dialogues (using standard RISC OS procedures). Certain general operations (saving, printing, and editing) appear on both the Worksheet and the Chart Menus. The most frequently used operations can also be chosen from the keyboard. These keystroke equivalents are shown on the screen menus and are listed in the Quick Reference Guide (see page 71).

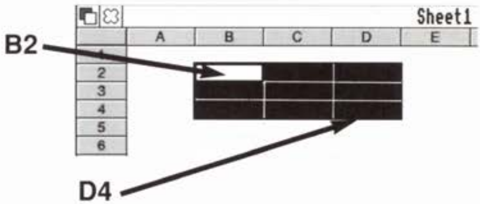
Creating a Spreadsheet

Cells, Data and Formulas

Addressing Cells

A worksheet consists of a grid of cells. Each individual cell is identified by its co-ordinates or **address**. The top left cell, for example, is A1; the cell at the intersection of column D and row 2 is D2.

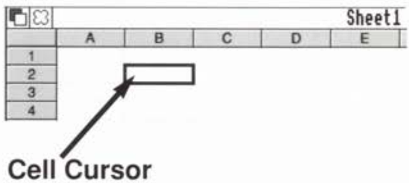




Some operations work on a **range** or group of adjacent cells: a range is identified by the co-ordinates of its two diagonally opposite corners. For example, the range that includes all cells between column B row 2 and column D row 4 is B2:D4. The two addresses are separated by a colon.

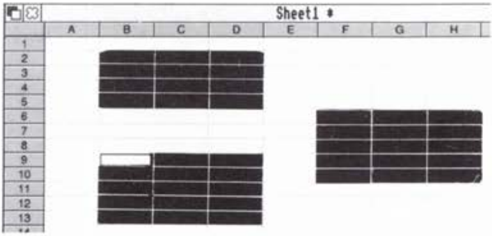
Selecting Cells

Before you can enter data, or perform any operation, you must indicate, or 'select' the cell (or cells) you want Eureka to use. The selected cell is known as the **active** cell.



To select a cell, position the worksheet cursor on it and press Select. The cell will be marked with a thick outline, called the cell cursor. To select a complete row of cells, click on the row number; to select a column, click on the column letter. To select several rows or columns, click on the first with Select and the rest with Adjust (the right mouse button).

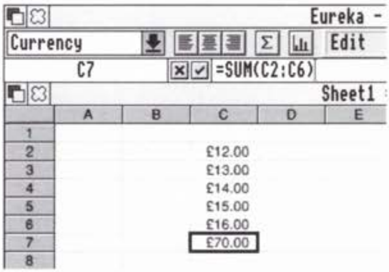
If you want to select a range of cells, move the cursor to one corner of the block, press and hold the left button, and 'drag' the mouse until the whole range is highlighted. Release the left button.



If you want to select several unconnected ranges of cells (for example, when style editing) press and hold the CTRL key, use the mouse to mark each range as described above, then release the CTRL key.

Entering Data

A cell can contain words, a number or a calculation. These are all types of data. To enter data into a cell:



• select the cell.

• enter the data via the keyboard. As you type, it will appear in the data entry field. To insert it into the selected cell, press Return.

You can correct or alter data after you have inserted it into a cell. Select the cell: its contents will appear in the data entry field. Press F2 to enter edit mode, then use the cursor keys and the backspace key. Click on ✓ to confirm or ✖ to abort the edit.

Entering Data into a Range of Cells

To enter data into a range of cells:

• select the range.

• type in the first value; press Return or Tab; type in the next value; and so on. You can also move backwards, using Shift Tab.

Note that when you select a block of cells, only one cell is actually 'active' and marked with the cell cursor. The cell cursor moves to the next cell as you enter each value.

Using Formulas

The essence of a spreadsheet like Eureka is its ability to perform ongoing calculations. The formula you insert into one cell uses data from other cells to perform a calculation and display the results. The process is instantaneous. If you alter the raw data, Eureka automatically performs the calculation again and displays the new answer (unless you have selected the Manual Calculation option, see page 17).

Entering a Formula

To enter a formula:

• select the cell.

• Type '='
(this tells Eureka that you are about to enter a formula).

• Enter the formula, referring to the data cells by their addresses.

For example:

=A1+A2 (add the contents of A1 and A2)
=A1-B1 (subtract the contents of B1 from A1)
=SUM(A1:A3) (sum the contents of A1, A2 and A3)

The formula will appear in the data entry field.

You can either type the cell addresses or 'point' to them - position the pointer on a cell, click, and the address will appear in the text field. If you want to point to a range of cells, position the cursor on the first cell, press and hold Select, drag the cursor to the last cell, release Select. The two addresses will appear in the text field.

• To insert the formula into the selected cell, press Return.

The formula will not appear in the cell; instead, the **results** of the calculation will be displayed.

(If the formula does appear in the cell, you have forgotten the = sign, and Eureka has treated the cell contents as text!)

If you want to examine the formula in a particular cell, select the cell, and the formula will appear in the data entry field. If you want to alter or correct it, press F2 to enter edit mode, then use the cursor keys and the backspace key. Click on ✓ to confirm or ✖ to abort the edit.

Relative and Absolute Referencing

A formula can be copied from one cell to another using Eureka's copy/paste facilities (see pages 18-19). When you copy a formula, any cell addresses are automatically adjusted. For example, if you copy

=SUM(A1:A3)

from A4 to C4, the formula will be altered to

=SUM(C1:C3).

This is called relative referencing.

However, if you want the formula to refer to the same address or addresses, even if it is copied to other cells, you can use an absolute reference.

To make a cell reference absolute, or partially absolute, simply insert a dollar sign before one or both parts of the cell address. For example, a formula containing

$A$3

will always refer to cell A3. A formula containing

$A3

will always refer to column A, though the row may be adjusted.

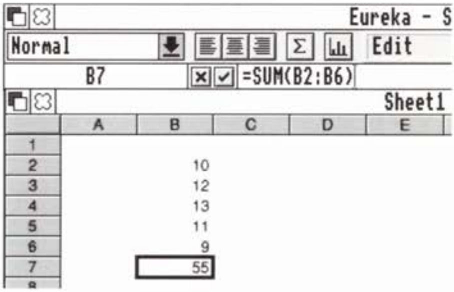
Similarly,

A$3

will always refer to row 3, though the column may be updated.

You can either type the dollar signs, or insert them by positioning the caret on the cell address and pressing F4 repeatedly to step through the various options. (If the caret is not on the cell address, you will hear a warning beep.)

Functions



Eureka provides over 150 functions, including

SUM (sum values in specified cells)
AVERAGE (calculate average of values in specified cells)
MAX (display highest value in a range of cells)
MIN (display lowest value in a range of cells)
INT (display value in specified cell as an integer)
COUNT (display the number of cells selected)
SQRT (square root).

All Eureka's functions are described in detail in the function reference section of this manual.

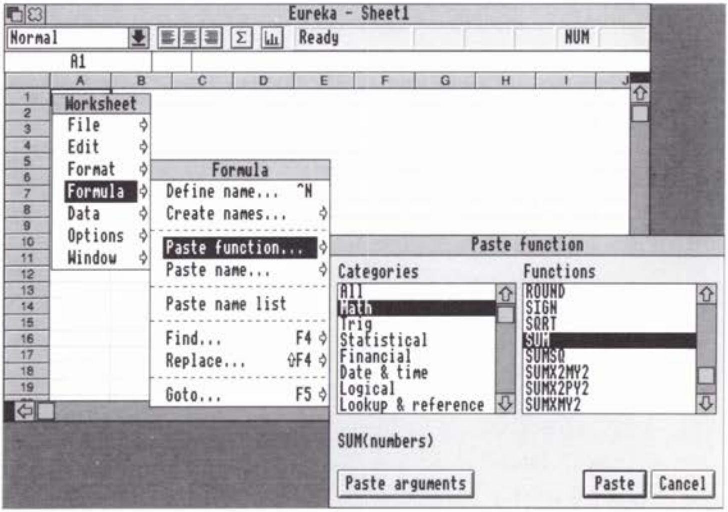
To insert a function into a formula:

• open Worksheet→Formula→Paste Function.

• To help you locate the function you want to use, the list is subdivided into 10 categories — Math, Trig, Statistical, Financial, and so on — then each category is listed in alphabetical order.

• Click on the appropriate category, and the subset of functions will be listed in the Functions window. (If you are not sure which category to select, click on All.)

• Click on the function you want. It will be displayed at the bottom of the dialogue, with a brief summary of the arguments it takes, for example:



• Click on Paste. The function will appear on the data entry bar.

Alternatively, you can type the name of the function directly into the data entry field.

Paste Arguments

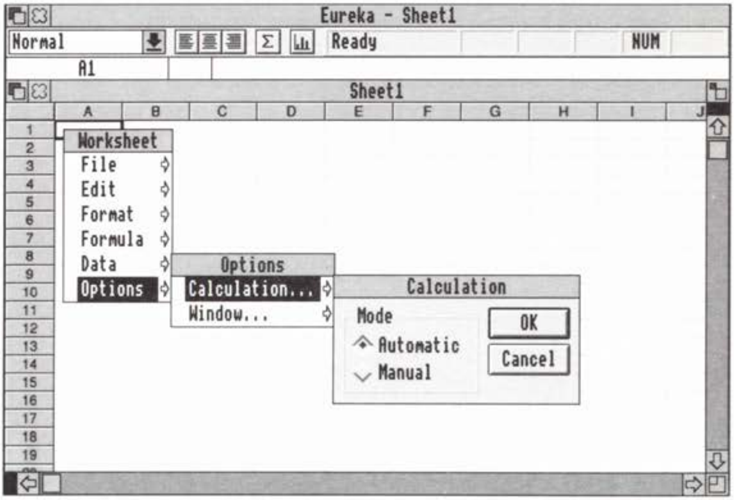
Some functions take very complex arguments:

IF (logical_test,result_if_true,result_if_false)

Eureka, therefore, allows you to use the argument format as a memory aid, by pasting it into the data entry field by clicking on Paste Arguments. (Remember to delete the format information as you enter the values.)

Manual Calculation

Eureka automatically recalculates formulas when you alter the relevant data. But in certain circumstances, you may prefer Eureka to wait until you request a recalculation.



To select manual calculation mode:

• Open Worksheet→Options→Calculation.

• Click on Manual.

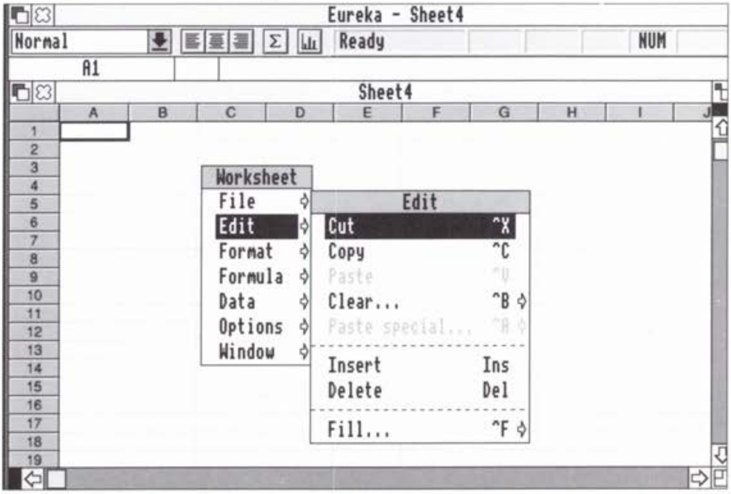
• Click on OK.

You can now control recalculation by either opening Worksheet→Options and selecting Calculate Now, or by pressing F9. If you alter data that affects a formula, Eureka displays 'CALC' in the status bar to remind you that a recalculation is necessary.

Editing a Worksheet:The Edit Menu

Eureka provides standard RISC OS editing facilities (plus some specialised spreadsheet facilities) via the Edit Menu.

To access the Edit Menu, open Worksheet→Edit.



As you become more familiar with Eureka, you will probably find it quicker to use the keyboard for editing. Keystroke equivalents are given on the Edit Menu and are also listed below.

Cut

Cut (^X) allows you to remove data from the active cell or cells and paste it elsewhere. Note that the data remains displayed in the original cells until you perform the paste.

Copy

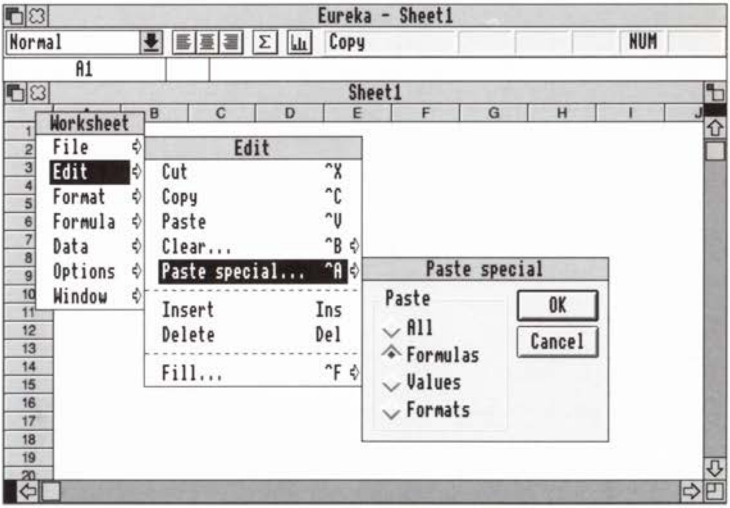
Copy (^C) takes a copy of the active cell or cells, and stores the data for pasting elsewhere. Copied data can be pasted repeatedly, so the highlighted boundary persists around the original data until you perform another Copy. If you want to deselect the data (and thereby erase the boundary) press ESC.

Paste

Paste (^V) is used to insert cut or copied data into the active cell or cells.

Paste Special

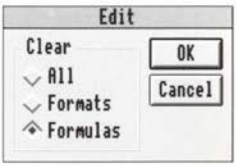
Paste special→(^A) provides four paste options:



All data, **Formulas** only, **Values** only, or **Formats** only. Select the option you want by clicking on the appropriate box, then click OK. The dialogue automatically returns to the default setting, Formulas.

Clear

Clear→ (^B) provides three options: **All** erases all data; **Formats** removes all formatting information but leaves the data in the cells; **Formulas** deletes formulas but leaves other data intact.



Insert

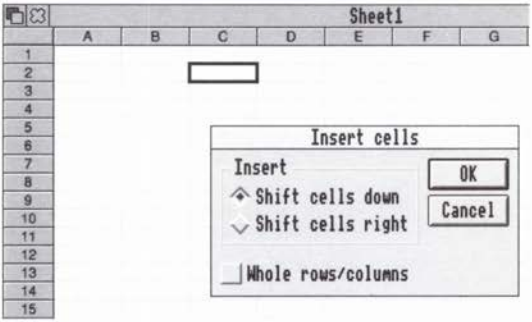
Insert (or the Insert key) allows you to insert rows, columns or individual cells into the worksheet.

To insert a row, click on the row number where you want to make the insertion; select Insert or press the Insert key.

To insert a column, click on the column letter where you want to make the insertion; select Insert or press the Insert key.

To insert several rows or columns, click on the first row or column with Select and the rest with Adjust. Open Worksheet ➔ Edit and select Insert (or press the Insert key).

To insert a cell, click on the cell where you want to make the insertion. Select Insert (or press the Insert key). The active cell can either be shifted down or to the right and you can opt to shift the whole row/column (in effect, insert a whole row or column) or just the cell. Click on the appropriate box, then click OK.

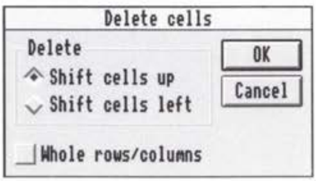


Delete

Delete allows you to delete rows, columns or individual cells from the worksheet.

To delete a row, click on the row number; select Delete or press the Delete key.

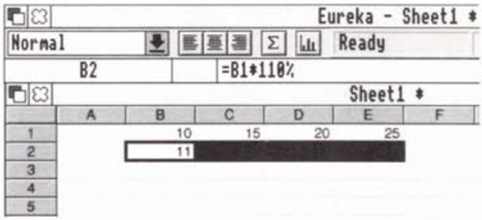
To delete a column, click on the column letter; select Delete or press the Delete key.



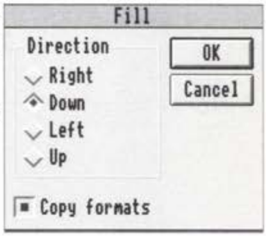
To delete several rows or columns, click on the first row or column with Select and the rest with Adjust. Open Worksheet → Edit and select Delete (or press the Delete key).

To delete a cell, click on the cell and select Delete (or press the Delete key). The gap can be filled either by moving the cell below upwards, or by moving the cell to the right leftwards. You can opt to move the entire row/column (in effect, delete the whole row or column), or just one cell - click on the appropriate box or boxes, then click OK.

Fill



Fill (F5) allows you to fill a row or column of cells with data from an existing cell. It is most often used for copying complex formulas. Highlight the cell you want to copy from and the cell or cells you want to copy to (they must be contiguous). Select Fill and choose Right, Down, Left, or Up as appropriate. In addition, you can opt to copy the format as well as the data. Click OK.



FORMATTING THE WORKSHEET

Eureka provides a number of DTP-type formatting facilities, so you can use outline fonts, lines and blocks of colour to transform basic worksheets into business stationery, company reports, and materials for presentations and publication.

The Align Buttons

The simplest of these are the align left, align centre and align right buttons, which allow you to position text and numbers within a cell or block of cells. Select the cell or cells and click on the appropriate button.

The Style Menu

The Style Menu, situated on the Control Bar, presents a number of predefined styles that you can apply to text or numerical data.

A style is a set of formatting instructions that has been saved with a descriptive name. To apply a style, select the appropriate cell or cells, open the Style Menu by clicking on the arrow, then select the style name. The data will be reformatted according to the style you have chosen.

The built-in styles available on the Style Menu are :

Normal - which sets the font to the basic worksheet font but does no special aligning or formatting of data

Comma - which automatically inserts commas in numbers over three figures

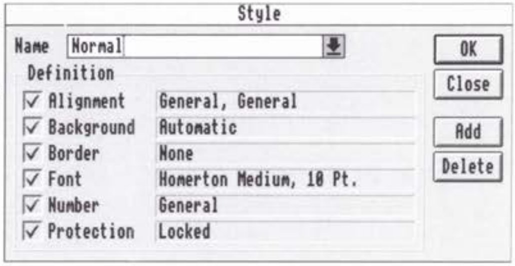
Currency - which adds a £ sign and trailing zeros to represent pence

Percent - which adds two trailing zeros and a % sign

Note that the Comma, Currency and Percent styles combine with Normal: they provide the numerical format, Normal supplies the font style and size.

Creating a Style

You can create customised styles and add them to the Style Menu, using the Style dialogue.



To create a style:

• either open Worksheet➔Format and select Style; or press ^F5.

• If you want to modify an existing style, click on the arrow to display the entire list, then click on the chosen name. It will appear in the name field.

• If you want to create a completely new style, use backspace to erase the existing name, then type in a new one.

• The style you are creating is to be applied to cells; so the style dialogue allows you to define:

data alignment (within the cell)
background colour
border style and colour
number format
protection level
font style, size and spacing

Not all of these attributes have to be defined; you can 'switch off' any that are not relevant by clicking on the appropriate radio button.

• To define an attribute, click on the current setting to display a dialogue. (The dialogues are discussed in the Format Menu section, below.)

• When you have finished defining the style, click on Add to add its name to the Style Menu.

Deleting a Style from the Style Menu

To delete a style from the Style Menu:

• open Worksheet→Format and select Style.

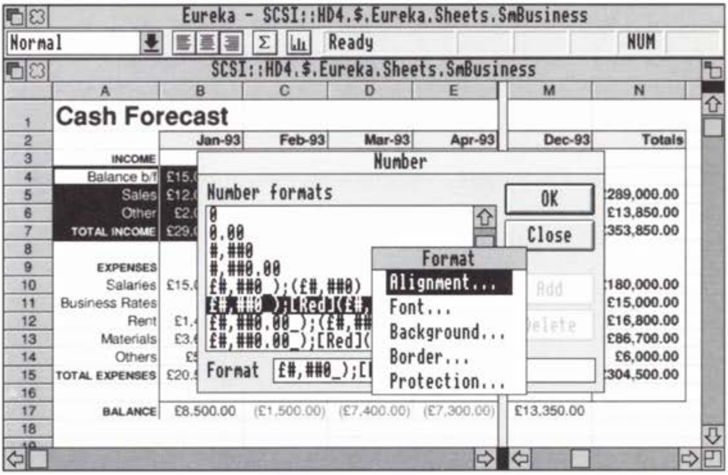
• select the name of the style you want to delete.

• click on the delete button.

Note that Eureka will not allow you to delete any of the predefined styles.

The Format Menu

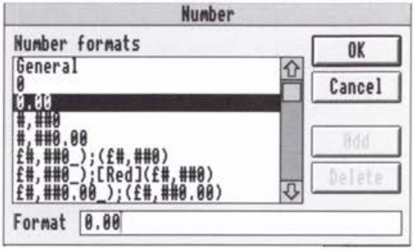
The Format Menu provides the same options as the Style dialogue but allows you to format cells on the screen, by altering one attribute at a time.



As you become more familiar with the individual dialogues and the attributes they control, you may find that you prefer to move directly from one dialogue to the next; you can do this by positioning the pointer anywhere on the background of the current dialogue and pressing Menu, then clicking on the appropriate option.

Selecting another dialogue is the equivalent of clicking on OK; any changes you have made via the first dialogue are immediately added to the currently selected cell or range, and cancelling the second dialogue does not affect them.

Number→(^F1)



The number dialogue provides over 20 predefined numerical formats, including

0.00 (format with two decimal places)
#.##0 (round up to two decimal places)

date formats, such as

d-mmm-yy (5 Jan 92)
d/m/yy (11/12/57)

and various currency options.

To apply a numerical format:

• select the cell or cells.

• open Worksheet→Format→Number.

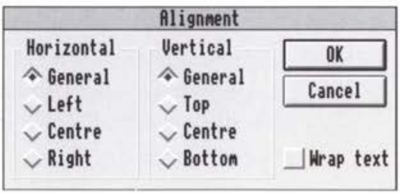
• click on the desired format. It will appear in the Format field and can be modified, if required.

• Click on OK.

If you create a new number format, or modify an existing one, you can add it to the list by clicking on Add. (This does not apply the format to the highlighted cells.) Number formatting is discussed in detail in the reference section of this manual.

Alignment→(^F2)

The Alignment dialogue provides a more precise mechanism for aligning data than the buttons on the Control Bar, by allowing you to specify both the horizontal and vertical position of the data.

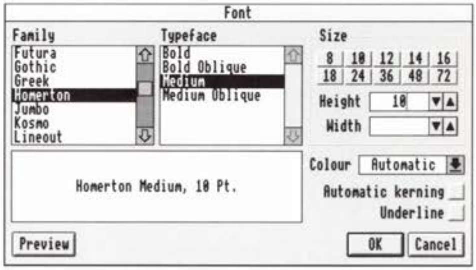


The Wrap text option defines how Eureka handles long strings of data (such as headings) that do not fit into the existing column width. If Wrap text is selected, the data is accommodated by increasing the number of lines in the cell; otherwise some of the data will be hidden unless you increase the column width.

Wrap text defaults to off.

Font (^F3)

The Font dialogue provides powerful facilities for formatting text in charts and tables. It allows you to select font, style (medium, bold, oblique), point size, character width (expressed in points) and text colour.



In addition, you can opt for automatic kerning (proportional spacing between letters according to letter shape) and text underlining.

• Select the cell or cells.

• open Worksheet→Format and select Font.

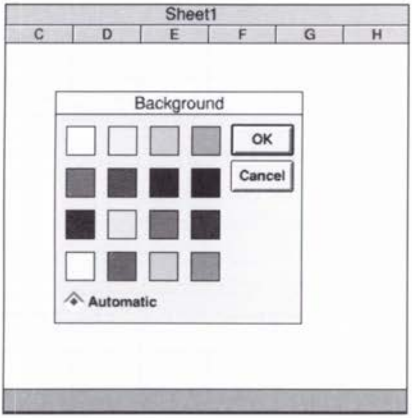
• select font, size, etc.

• If necessary, you can check typeface style and size (before applying it to your worksheet) by clicking on Preview; the font name appears, drawn to size, in the preview window.

• Click on OK.

Background (^F4)

Background fills the background of the selected cell or cells with a chosen colour.



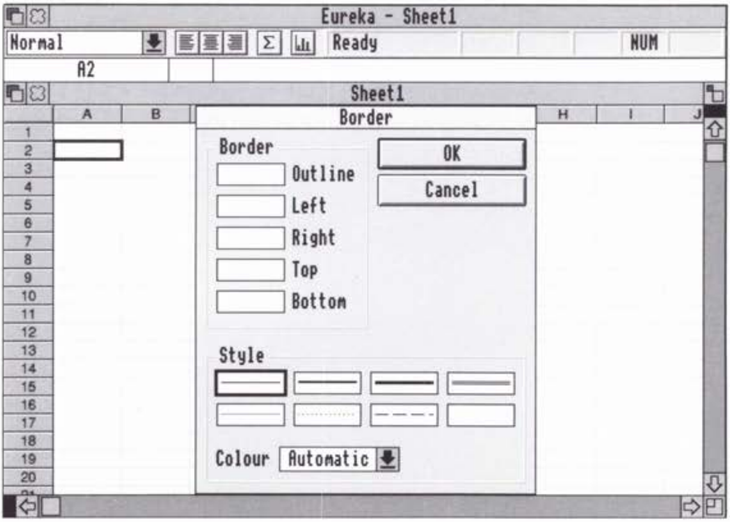
- Select the cell or cells.
- open Worksheet→Format and select Background.
- Click on the chosen colour.
- Click on OK.

Border (^SHIFT F1)

Border allows you to improve the presentation of your worksheet by drawing lines and borders. These are built up by outlining or partially outlining individual cells in a choice of line styles and colours.

To add a border to a cell or cells:

- select the cell or cells.



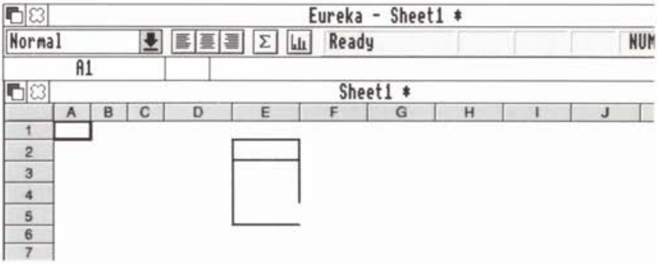
• open Worksheet➔Format and select Border.

• select line colour by clicking the arrow to display the range available, then clicking the desired colour.

• select desired line style.

• click on left, right, top, bottom, or a combination; or click on outline.

• click OK.



For example, the lines in the illustration have been drawn by

• selecting cell E2 and choosing Outline

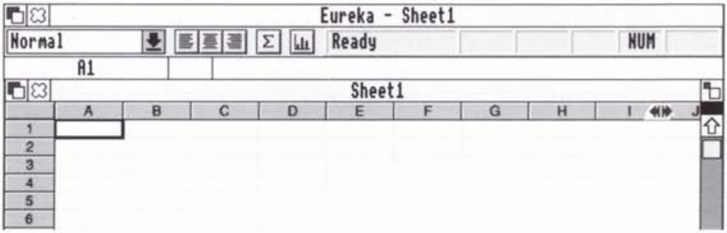
• selecting cells E3 and E4 and choosing Left and Right

• selecting cell E5 and choosing Left and Bottom

Note that if you select a **block** of cells, the border format is applied to each individual cell, unless you select Outline, which draws a line around the block.

Altering Row and Column Widths

When Eureka opens a worksheet, all rows are set to the same height and all columns to the same width. However, once you have entered data, you may need to reformat the rows and columns to more appropriate sizes.



Using the Mouse

The easiest way to alter the size of a row or column is with the mouse:

- Move the cursor onto the label bar and position it on the gridline you want to alter.

- the cursor will change to a double-headed arrow.

- press Select and drag the gridline to the new position.

- release Select.

You can use the same method to alter several rows or columns:

- select the rows or columns you want to change by clicking the first with Select and the rest with Adjust.

- place the cursor on the label bar and alter one row or column.

- all the highlighted rows or columns will be redrawn to the same size.

You can also 'auto-size' a column or columns to fit the longest piece of data by double clicking on the gaps between the column labels.

Column width→(^SHIFT F3)

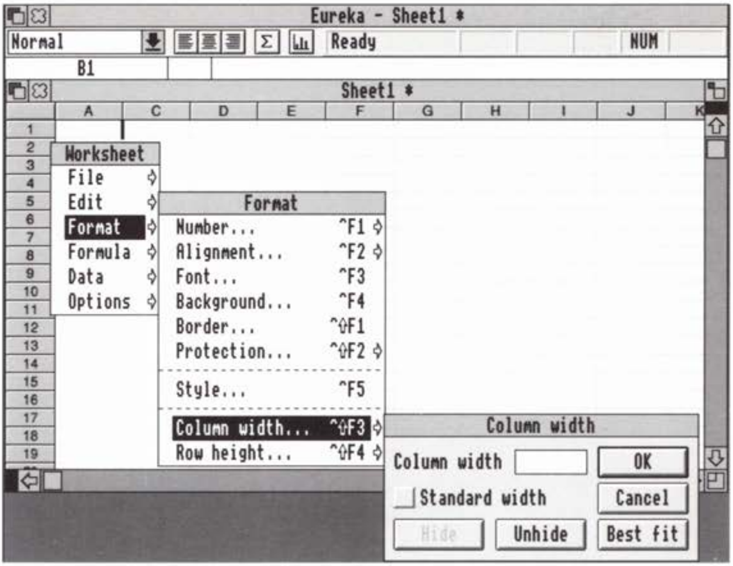
In addition, columns can be sized via the Column width dialogue:

- Select the column or columns.

- open Worksheet→Format→Column width or press ^SHIFT F3.

- the current column width is displayed: (1 unit = the width of a 0 in normal style) use the cursor keys and the backspace key to alter the value.

- Click on OK.



Alternatively,

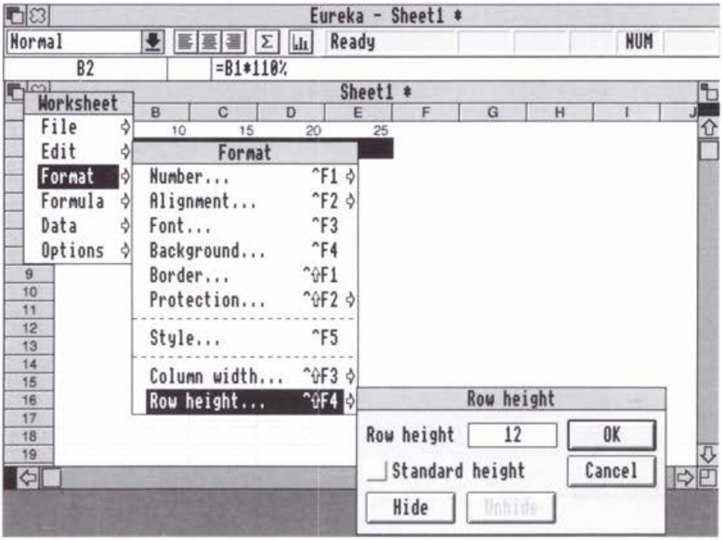
• Clicking on Standard width resets the column to its original size.

• Clicking on Best fit sets column width to accommodate the longest piece of data.

• Clicking on Hide/Unhide temporarily removes/replaces the column.

Row height→(^SHIFT F4)

Rows can be sized via the Row height dialogue:



• Select the row or rows.

• Open Worksheet→Format→Row height or press ^SHIFT F4.

• the current row height is displayed; use the cursor keys and the backspace key to alter the value.

• Click on OK.

Alternatively,

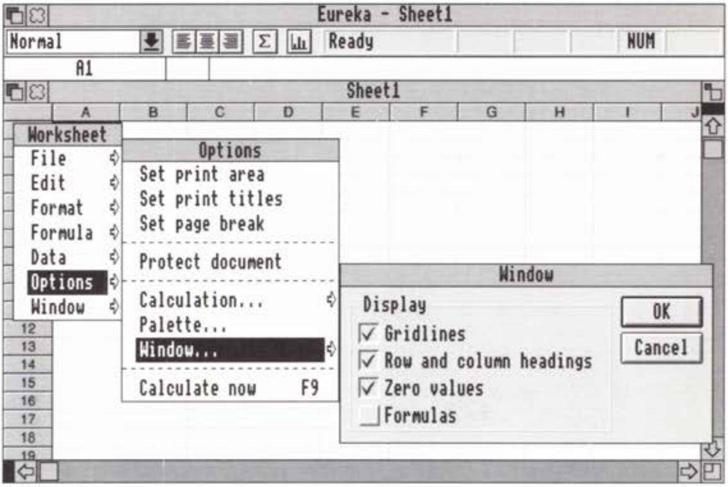
• Clicking on Standard height resets the row to its original size.

• Clicking on Hide/unhide temporarily removes/replaces the row.

View Options

Display Options

If you prefer, you can turn off Eureka's gridlines and row and column labels:



• Open Worksheet ➔ Options ➔ Window.

• Click on the appropriate radio buttons.

• Click OK.

The settings are saved when you save the worksheet; next time you open the worksheet, it will be displayed according to these settings.

Zero Values

If the cells used by a formula are empty, the result of any calculation may be zero. (This may occur on a worksheet that has been formatted as a standard form - an invoice, for example - when only some of the rows/columns have been filled in.) If, under these circumstances, you want Eureka to display empty cells.

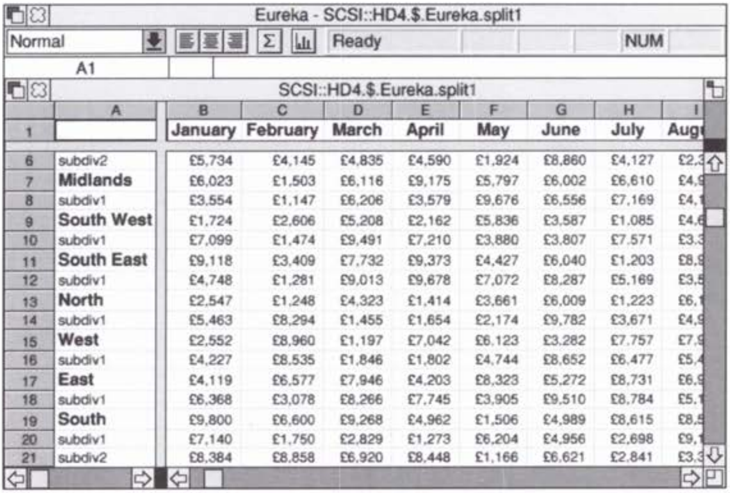
• Open Worksheet ➔ Options ➔ Window.

• Deselect zero values.

• Click on OK.

Viewing different parts of the worksheet simultaneously

It can be frustrating to scroll back and forth across a large and complex worksheet. Eureka allows you to split the worksheet window into several independently scrollable areas, so you can display opposite ends of the sheet simultaneously. (This is simply for the purpose of viewing the worksheet on the screen; it does not alter its format.)



To split the worksheet window horizontally, or vertically:

• position the cursor on a splitter (the black band at the top right and the bottom left corners of the window.)

• press and hold Select, and drag the blue band into position.

• Remove a splitter by double clicking on the black band.

New View

An alternative to splitting a worksheet is to display several copies of the same window, each copy showing a different part of the sheet.

To create another copy of the current worksheet, open Worksheet ➔ window and select Window ➔ New View. The copy will be displayed in front of the old version and will be active (the pointer will be inside the new window and all operations will be applied to that worksheet until you move the pointer to another sheet).

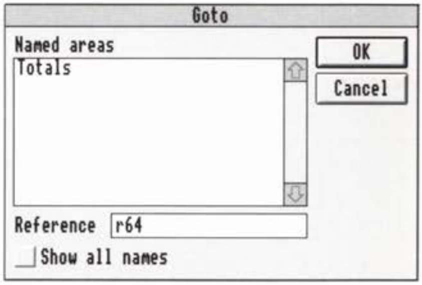
Goto

The Goto facility allows you to move quickly to a specified cell without scrolling.

• Open Worksheet ➔ Formula ➔ Goto.

• Type the cell address into the Reference text field.

• Click on OK.



Hiding and Un-hiding Windows

You can temporarily close a worksheet by opening Worksheet ➔ Window and selecting Hide.

To re-open the worksheet:

• position the pointer on the Eureka icon on the icon bar.

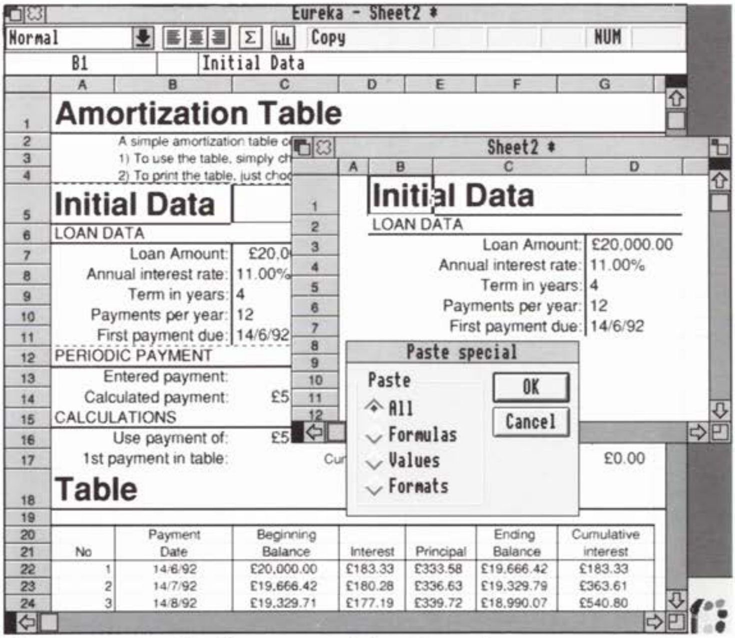
• Press Menu and select Unhide.

• Eureka will display a list of hidden worksheets - click on the one you want to open.

Using Multiple Worksheets

Eureka allows you to combine information from different worksheets, copying data, formulas and formatting information from one sheet to another.

Moving or copying Data



To move or copy data from one worksheet to another:

• open both worksheets.

• Select the cells you want to copy data from.

• Making sure that the pointer is still in that worksheet, open Worksheet → Edit and select Cut (to move) or Copy.

• Move the pointer into the other worksheet and select the cells you want to move or copy the data to.

• Making sure the pointer is in that worksheet, select Paste or Paste special (the latter allows you the option of pasting all data, formulas only, values only, or formatting information only).

Linking Worksheets

You can also create links between worksheets. For example, a formula in one worksheet can refer to a cell on another, and the cell address (which in this case includes the worksheet name, directory path, etc.,) can be inserted by pointing, in the usual way. (Linked worksheets are described in detail on page B - 35 in the reference section of this manual.)

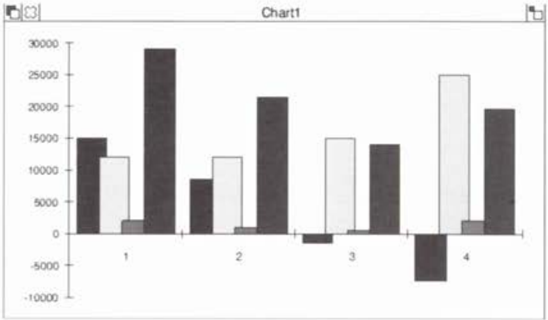
Creating Graphs and Charts

Eureka supports a comprehensive range of charting facilities which rapidly turn your data into graphs and charts suitable for reports or presentations.

To create a chart:

• open a worksheet.

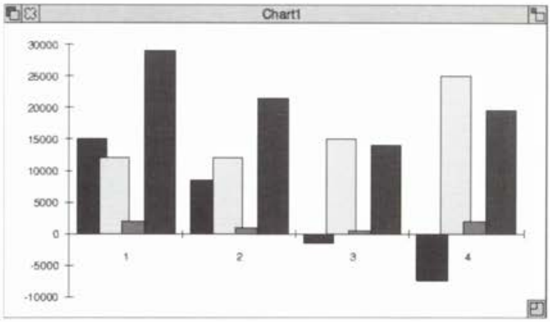
• indicate the data you want to use by selecting the appropriate cells.



• either open Eureka ➔ New document ➔ Chart or click on the Chart button, on the Control Bar.

• Eureka will open a Chart window, displaying the selected data as a bar chart.

Re-sizing the Chart Window



The Window Sizer icon does not normally appear in the Chart window, but is displayed on demand. To re-size the Chart window:

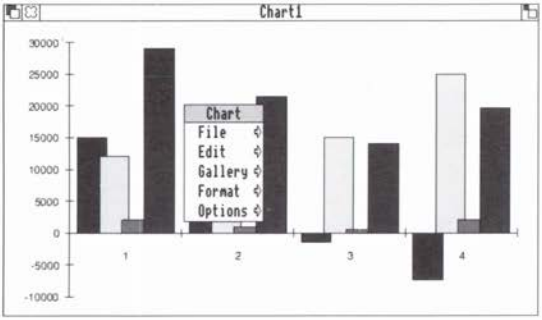
• place the pointer on its lower right-hand corner.

• press and hold Select. The Window Sizer icon will appear.

• drag the corner to the new position, then release Select.

The Chart Menu

You can choose different chart styles, and format and label basic charts, via the Chart Menu,which is opened by positioning the pointer anywhere in the Chart window and pressing Menu.



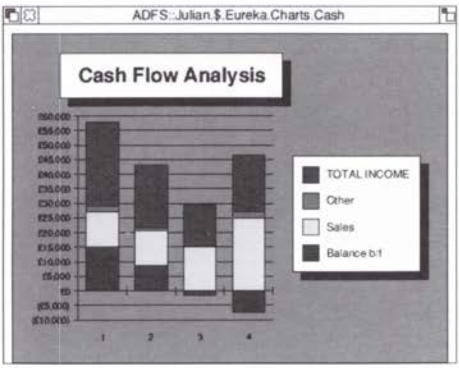
Choosing a Chart Style: Gallery

The Gallery Menu presents a range of chart styles - Area, Bar, Column, Line, Pie and Scatter. To apply one of these styles:

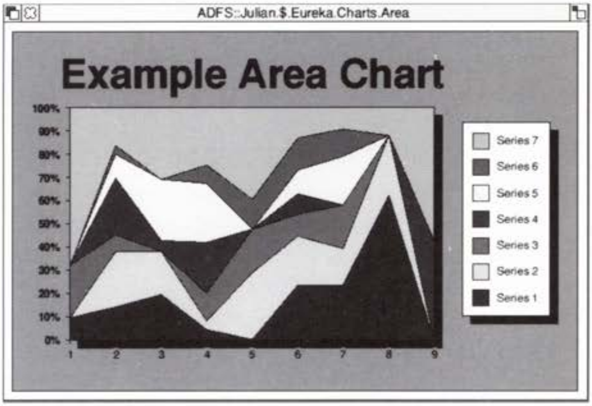
- open Chart ➔ Gallery and click on the appropriate menu item.

- In addition, Eureka provides three further display options:

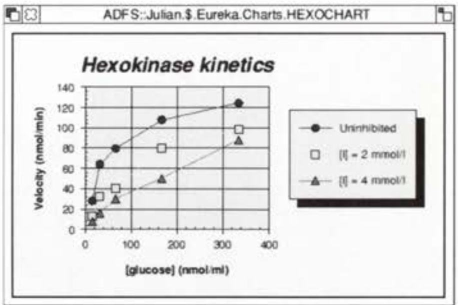
Certain types of chart - column, bar, area and line - can be displayed 'Stacked', with different sets of data drawn one on top of the other.



The Percentage option converts the data to values between 0 and 1, for display as a percentage chart (if you choose this option, you will also need to re-format the axes, using the Number dialogue described below, to a percentage format).



Scatter graphs can be displayed with a connecting line between the points.



These options only become available when you select the appropriate chart style, to apply them, choose the basic chart style using Adjust, then click on the further option with Select.

Displaying the Legend

When you create a chart, Eureka automatically compiles a Legend, or key, to the data. This is not displayed by default, but can be displayed via the Legend dialogue.

• Open Chart ➔ Options ➔ Legend.

• indicate where you want the legend to appear - Right, Bottom, Left or Top. (Selecting None switches the Legend off.)

• click OK.

If the data you selected when you created the chart included text labels in the topmost and leftmost cells, these will have been used to compile the Legend; if not, the data will be labelled 'Series 1, 'Series 2', and so on.

Formatting the Chart: Format

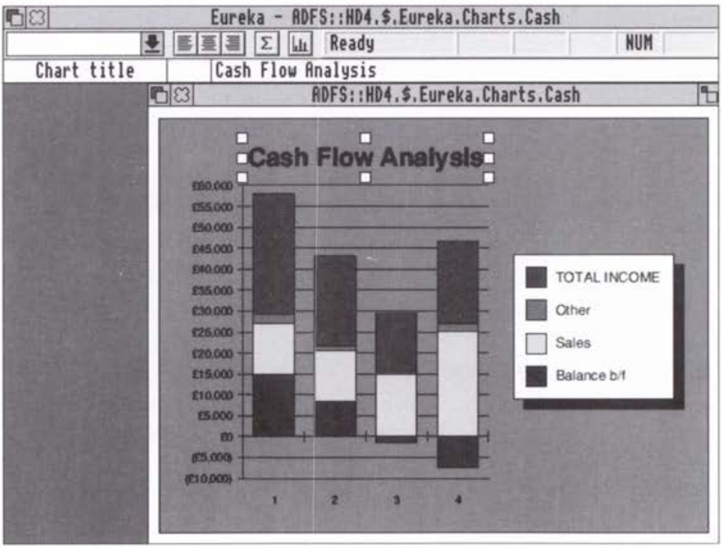
Having created a basic chart in the chosen style, you can improve its presentation using the Format option, which allows you to access the Pattern, Font, Text, Number and Scale dialogues.

Selecting Chart Elements

Before you can open any of the formatting dialogues, you must first indicate which element of the chart you want to modify. To select part of a chart:

• position the pointer on the element and press Select.

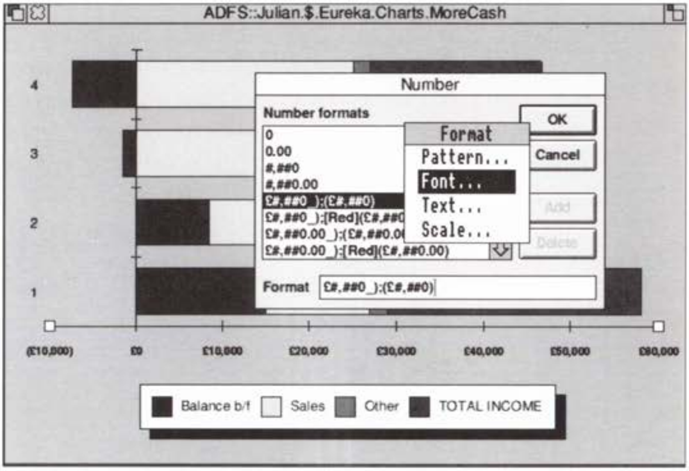
• Handles will appear on the element you have selected, and its name will be displayed in the data entry bar. You can then open Chart → Format and



select the dialogue you want. (If the dialogue you choose is not applicable to the selected element, Eureka will not allow you to open it.)

Moving Between Dialogues

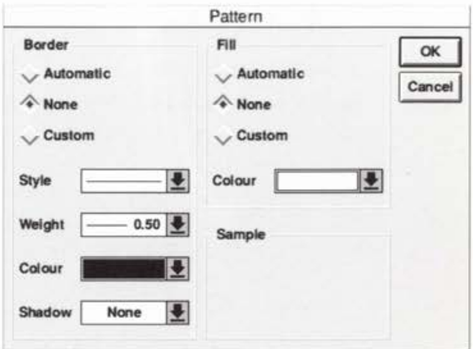
You can move from one dialogue to another by positioning the pointer anywhere on the background of the current dialogue and pressing Menu, then clicking on the appropriate option.



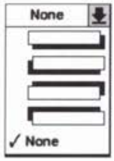
Selecting another dialogue is the equivalent of clicking on OK; any changes you have made via the first dialogue are immediately applied to the chart, and cancelling the second dialogue does not affect them.

Pattern

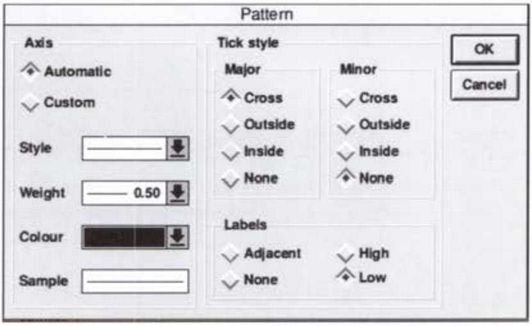
The Pattern dialogue controls the style of the various elements of the chart. The contents of the Pattern dialogue will vary according to what type of element is selected - chart background, plot area, axes, legend or data.



If you have selected the chart background, plot area, or legend, the Pattern dialogue will allow you to modify the Border (which otherwise defaults to off), choosing line style, line weight, colour (remember to select the colour before any of the other options). You can also apply different styles of shadow - click on the arrow to display the complete range available - and/or a fill colour.



If you have selected an axis, the Pattern dialogue allows you to modify line style and tick style and to adjust the position of numerical labels. Ticks are the marks that denote points on the axes. Major ticks indicate major divisions - like whole numbers, tens, or hundreds. Minor ticks divide the space between major ticks. You can position the tick marks across the axis, or outside or inside the axis line, or can opt to display None.



As soon as you alter a setting on the Pattern dialogue, the 'Custom' option is automatically selected. If you want to cancel the new setting, just click on 'Automatic' - to restore the default - or 'None'.

Eureka displays the effect of certain options in a Sample field at the bottom of the dialogue. This allows you to keep track of the changes you have made and to 'preview' them before they are applied to the chart.

Adding Text

To add a text label to a chart, an axis, or data:

• select the element you want to label by positioning the pointer on it and clicking Select.

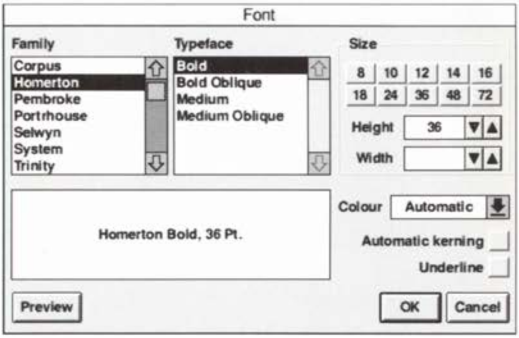
• Type in the text. It will appear on the data entry bar.

Once text has been attached to the chart, its style and background colour can be altered using the Font and Text dialogues.

Font

The Font dialogue allows you to modify the font, style (medium, bold, oblique), point size, character width (expressed in points) and text colour, of attached text, legend text and numerical labels. It is identical to the Worksheet Font dialogue.

Having selected the appropriate axis, plot area or legend box:



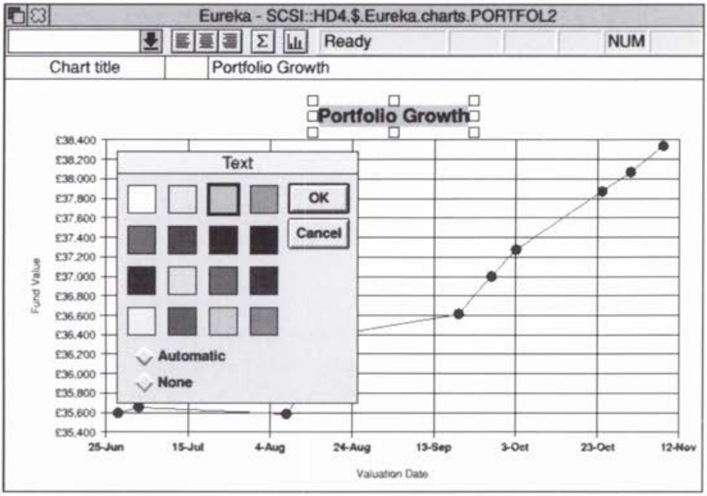
• open Chart ➔ Format and select Font.

• select font family, typeface, size, colour, etc.

• If necessary, you can check typeface style and size (before applying it to your chart) by clicking on Preview; the font name appears, drawn to size, in the previous window.

• Click OK (or press Menu and select another dialogue).

Text



The Text dialogue controls the background colour of text attached to the selected chart element.

• Select the appropriate chart element.

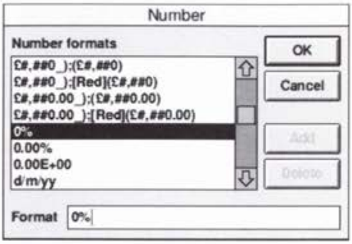
• open Chart ➔ Format and select Text.

• select the desired colour.

• Click on OK (or press Menu and select another dialogue).

Number

The Number dialogue provides over 20 predefined formats, including percentage and currency, that can be applied to numerical labels attached to the selected element. It is identical to the Worksheet Number dialogue.



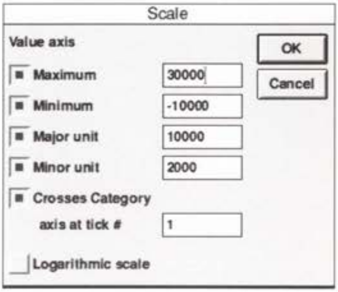
• Select the appropriate axis or plot area.

• open Chart ➔ Format and select Number.

• click on the desired format. It will appear in the Format field and can be modified, if required.

• Click on OK (or press Menu and select another dialogue).

Scale



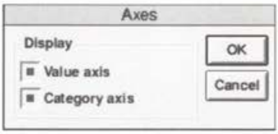
The Scale dialogue allows you to edit the axes and manually alter scale. Its contents vary according to the contents of the chart and which axis is selected.

If you select the value (vertical) axis, the Scale dialogue allows you to alter maximum and minimum value (so if all your data lies, for example, between 50 and 60, you can change the default scale (0-60 to 40-60); set major and minor units (as indicated by the tick marks) and specify where this axis crosses the Category (horizontal) axis. You can also opt for a logarithmic scale.

If you try to open this dialogue when an element other than an axis is selected, Eureka will give you a warning beep.

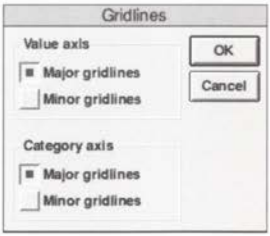
Displaying Axes and Gridlines: Options

You can switch either or both axes off (and on again), by:



• opening Chart ➔ Options and selecting Axes. (It is not necessary to select an axis before opening the Axes dialogue.)

• The axes are identified as Value (vertical) and Category (horizontal).



• Click the appropriate radio button or buttons.

• Click OK.

Similarly, you can display major and minor gridlines on either or both axes by:

• opening Chart ➔ Options and selecting Gridlines.

• The axes are identified by name, and you can opt to display gridlines on one, both or neither. Click the appropriate radio button or buttons.

• Click OK.

Saving A Worksheet or a Chart

To save a worksheet or a chart:

• open Worksheet ➔ File and select Save or open Chart ➔ File and select Save.

• If the file has not been saved before, the Save as dialogue will appear. Use the mouse and the backspace key to delete the default name ('Sheet1' or 'Chart1', etc) and type in a filename, up to 10 characters long. (Certain characters, including $, should not be used in filenames - see the Acorn Reference Manual for details.)

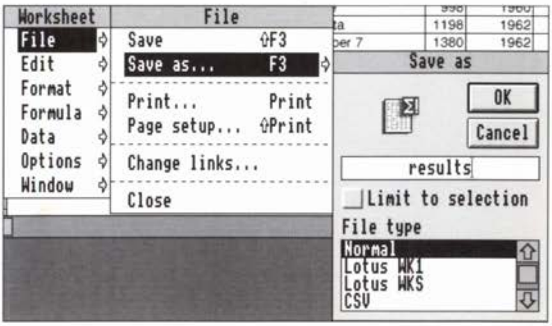
• Position the mouse on the icon, press Select, drag the icon into a directory viewer, and release Select.

• When you are saving a worksheet, the Type field allows you to specify whether it is saved as a normal Eureka file, in a Lotus 123-compatible format, as a draw file, or as a CSV file. (These options are covered in more detail in the reference section of the manual.)

When you are saving a chart, the Type field allows you to specify whether you want it saved as a normal chart file or as a Draw file which can be imported into other software packages as an image.

• To abort the saving process, just click on cancel. The Save as window will disappear.

If the chart or worksheet has already been saved, Eureka will save it with its existing name, overwriting the previous version.



Save as

If you want to preserve the old copy of a chart or worksheet, or create a copy you can modify, you can save a new version by giving it a different name.

• Open Worksheet ➔ File and select Save as or open Chart ➔ File and select Save as.

• Type in a new name, then drag the icon into a directory viewer.

• Click the "limit to selection" radio button to simply save the current selection rather than the whole sheet.

Loading a Worksheet Chart or a Lotus 123 File

To load an existing worksheet or chart, either

• drag the file icon to the Icon Bar and drop it on the Eureka program icon, or

• drag the file icon onto the Eureka Control Bar.

Alternatively, you can

• double click on the worksheet or chart icon.

To load a Lotus 123 file, either

• drag the file Icon to the Icon Bar and drop it on the Eureka program icon, or

• drag the file icon onto the Eureka Control Bar.

Opening a New Worksheet

To open a new worksheet, either

• open Menu ➔ New document and select Worksheet, or

• click on the Eureka icon on the Icon bar

The new worksheet will appear on the screen, in front of any existing worksheets.

Opening a New Chart

Unlike a worksheet, you cannot open an 'empty' chart. To create a new chart, you must first open a worksheet and select some data. Then either,

• open Menu ➔ New document and select Chart, or

• click on the Chart icon on the Control Bar.

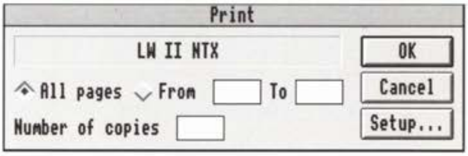
A Chart window will appear on the screen, in front of the worksheet, displaying the selected data as a bar chart. The chart can be modified, saved and printed via the Chart Menu.

Printing

To print a worksheet

• Ensure that your printer is on line.

• Open Worksheet➔File➔Print.



Select All if you want to print the entire worksheet; otherwise click on From and enter page numbers.

• Enter number of copies.

• Click on OK.

To Print a Chart

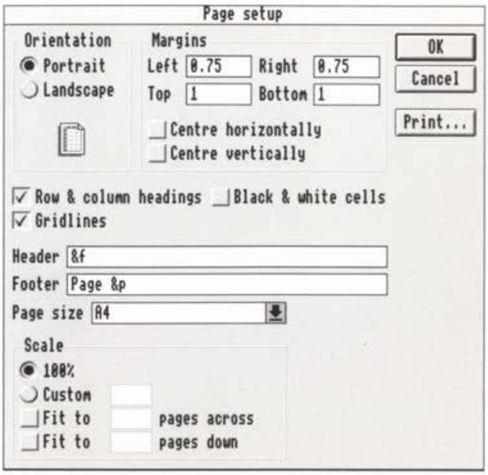
• Ensure that your printer is on line.

• Open Chart ➔ File ➔ Print.

• Click on OK.

The Page Setup Window

The page Setup Dialogue can be accessed from the Print Dialogue by clicking the Setup button, or by opening Worksheet ‘ File ‘ Page setup, or Chart ➔ File ➔ Page Setup. It allows you to:



• set paper orientation: select either portrait (vertical) or landscape (horizontal). The default is portrait.

• set left, right, top and bottom margins in mm and inches, or opt to centre the worksheet on the page, horizontally and vertically (provided the worksheet is smaller than the page).

• switch off row and column headings (1, 2, 3... and A, B, C...) and gridlines. The default is on.

• black and white cells.

• add a header or footer to every page of the worksheet. These default to filename and page number, respectively. Headers and footers use the normal style. You can change the font used by editing the normal style.

You can access the Print Dialogue by clicking on the Print button.

Black and White Cells

The Black and white cells option allows you to control the way a Worksheet with colour formatting is printed on a black and white printer. If the black and white cells option is off, Eureka converts the colour image into a grey scale image. If the option is on, Eureka prints the image in just black and white.

For more information on the page setup dialogue, see page B-50.

Quitting Eureka

To quit Eureka, open the Eureka Menu and select Quit. If the current worksheet or chart has been modified since you last saved it, Eureka will ask whether you want to Save, Discard or Cancel.

The Close Icon

As an alternative to quitting Eureka, you can temporarily close it down, by clicking the close icon on the Control Bar. This closes the Control Bar and all current worksheets/charts - but leaves the Eureka icon installed on the icon bar.

Closed files can be re-opened by positioning the pointer on the Eureka icon and pressing Menu to display the Eureka Menu, opening window, and selecting the worksheet or chart name.

You can also close an individual worksheet or chart by clicking on its close icon. If the file has been modified since you last saved it, Eureka will ask whether you want to Save, Discard or Cancel.

The current worksheet or chart can also be closed by opening Worksheet → File and selecting Close, or Chart → File and selecting Close. This method is particularly useful when you are displaying more than one view of a worksheet.

Using the keyboard
(in preference to the mouse)

If you would rather use the keyboard than a mouse, Eureka has been designed, as far as possible, to allow you to control most operations with keystrokes.

The cell cursor can be moved using the cursor keys: position it on the cell you want to activate and enter data in the normal way.

• If you want to select a range of cells, press F8 to enter 'EXTended mode'. When you move the cursor you will leave a trail of highlighted cells behind.

• Pressing SHIFT F10 extends the current range to encompass complete rows; pressing ^ F10 extends it to entire columns.

• Pressing F10 selects the entire worksheet.

• If you want to select more than one range of cells, press F8, select the first range; press SHIFT F8, move the cell cursor to the start of the next range; press F8, select the next range, and so on.

Once you have selected a range of cells you can move within it, using

• Return and ^Tab to move the cell cursor down one row (or back to the top of the range when you reach the bottom);

• Tab to move the cell cursor to the right (or back to the beginning of the range);

• SHIFT-Tab to move it to the left (or back to the end);

• ^SHIFT-Tab to move it up one row (or back to the bottom).

If you have selected several ranges, these keystrokes will move you from one range to the next.

In addition, Home moves the cell cursor back to cell A1, unless scroll lock is selected, when it moves the cell cursor to the top left cell.

The keystrokes that open menus and dialogues are given in the Quick Reference Guide (beginning on page 71).

Examples

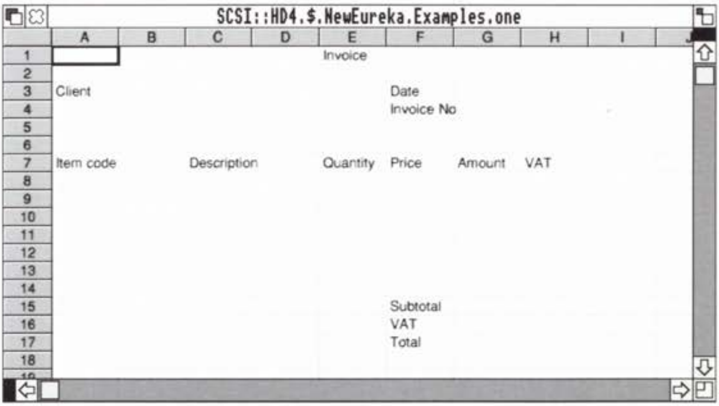
The examples that follow take you step-by-step through the creation of an invoice, a set of accounts and a cashflow forecast. They are intended to illustrate how a self-employed person or a small company might use Eureka in the day-to-day running of their business, and to provide you with a starting point for creating your own spreadsheets. These files can be found on the examples disc in your Eureka pack, inside a folder called 'Examples'.

Invoice

• Open a new worksheet.

• Enter the labels as shown in the diagram: select each cell in turn, type the text into the data entry bar, then press Return.

• If you make a mistake: re-select the cell; the text will be displayed on the data entry bar. Press F2, and the caret will appear; use the backspace key to correct the text.

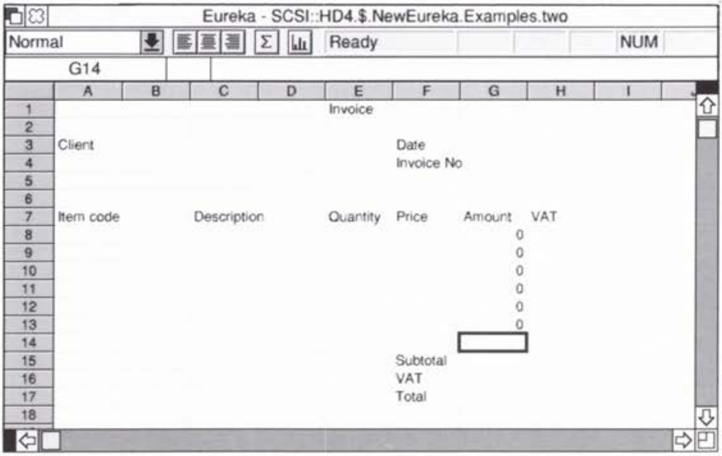


• Enter the formula =E8*F8 into cell G8. A zero will appear in G8

• This formula can then be copied into cells G9 to G14.

G8 is already selected, either press ^C or open Worksheet ➔ Edit and select copy.

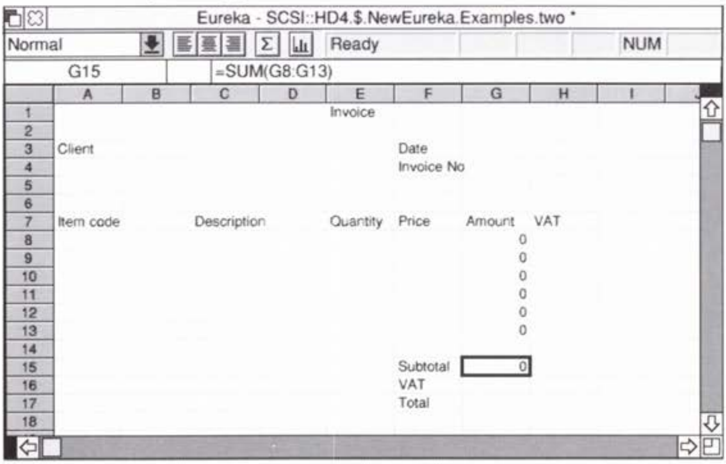
• Select cells G9 to G14. Either press ^V or open Worksheet ➔ Edit and select Paste. Zeros will appear in cells G9 to G13.



• Try selecting one of these cells. The formula it contains will appear in the data entry field; you'll notice that, because you have used relative references, the formula has been adjusted.

• To calculate the subtotal, select cell G15 and enter the formula =SUM(G8:G13)

or you can insert the cell addresses by pointing at the cell G15 and type? Then, using the mouse, position the pointer on cell G8, press Select, drag the pointer to cell G14, release Select. The addresses will have appeared on the data entry bar. Type in the final bracket.



• To calculate the VAT column, enter the following formula into cell H8 =G8*17.5%

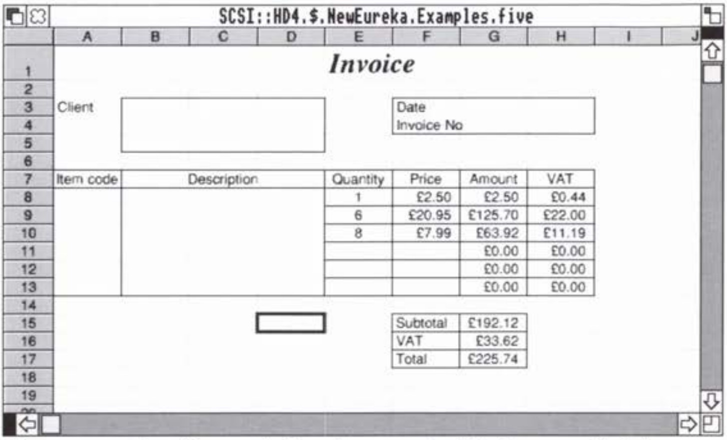
Then copy the formula to cells H9 to H13. This invoice assumes that all your products are rated at 17.5% VAT.

• To calculate the VAT total, enter the formula =SUM(H8:H13) into cell G16.

• To calculate the grand total, enter the formula =G15+G16 into cell G17.

Try entering some data into the Quantity and Price columns: do the values that appear in the Amount and VAT columns look like money? If not, select the range F8:H13; move the pointer up to the Control Bar; click on the arrow to open the Style Menu; and click on currency. Do the same with cells G15:G17.

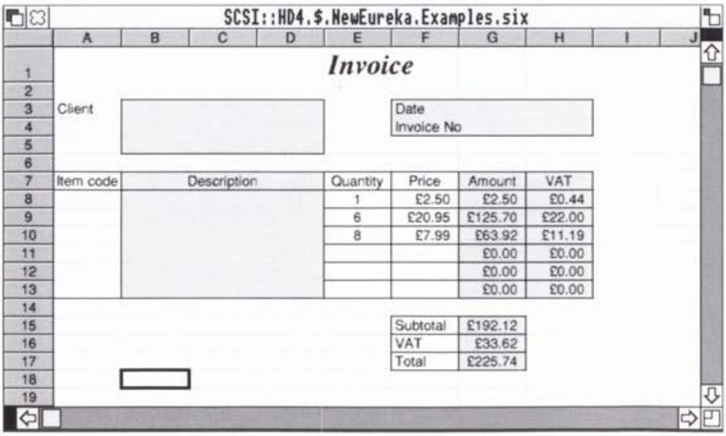
Where you have not inserted data into the Quantity and Price columns, the results of the calculations in columns G and H (Amount and VAT) are zero, and Eureka automatically displays £0.00. If you want these cells left empty, open Worksheet ➔ Options ➔ Window and deselect Zero values by clicking the radio button.



The next step is to format the invoice, to make it look more official.

• Select cell E1 - which contains the word 'Invoice'. Choose Format and select Font. Choose a clear typestyle (you may also want to choose bold or bold italic) and a reasonably large point size. Click on Preview to assess your choice. If you are happy with it, click OK.

• Format the rest of the worksheet, using the Border dialogue (^SHIFT F1 or Worksheet ➔ Format ➔ Border) to draw lines and columns. For example, select cells B3:D5, either press ^SHIFT F1 or open Worksheet ➔ Format ➔ Border and select Outline, then click OK. Select cells E7 to H13, open the Border dialogue and select Left, Right, Top and Bottom.



Remember that you can select a number of unconnected blocks, by holding the CTRL key whilst you make the selections, then apply the same format to all the blocks.

You may also want to use the Background dialogue (^F4 or Worksheet ➔ Format ➔ Background) to apply a background tint to some of the boxes to highlight important information.

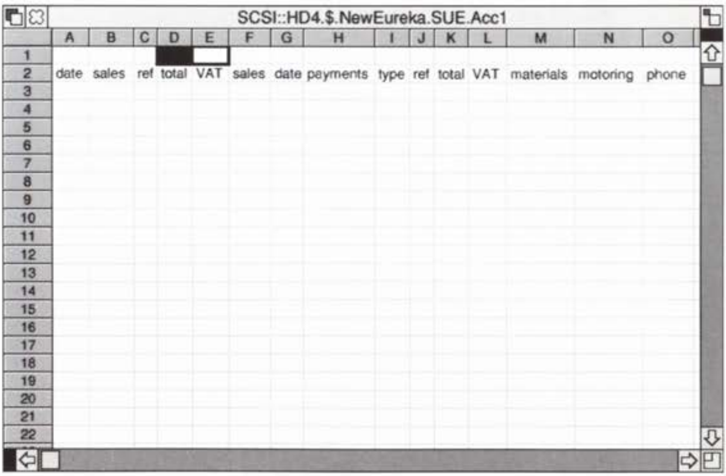
If you make a mistake, you can undo the formatting (but leave data and formulas unaffected) by selecting the appropriate cells, opening Worksheet ➔ Edit ➔ Clear and selecting Format, then clicking OK.

Accounts

• Open a new worksheet.

• Type in the labels as shown in the diagram: select each cell in turn, type the text into the data entry bar, then press Return. (The way you decide to break down your payments will depend on the nature of your business - the labels you will want to use in columns M - O may, therefore, be very different from this example.)

• If you make a mistake: re-select the cell, press F2, and use the backspace key to correct the text. Column B will be used to record customer names, etc; column H will be used to record purchases. To widen these columns to allow for text, position the cursor on the label bar, on the line between



column B and column C. The pointer will turn into a double headed arrow. Press Select and use the mouse to drag the gridline until it lies over the line between columns D and E; release Select. Widen column H in the same way.

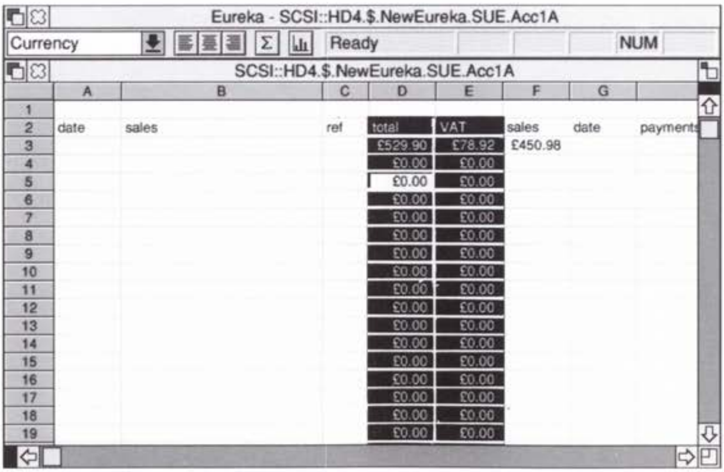
If you are VAT registered, you will need to keep separate records of sales without VAT(Sales), VAT, and sales inclusive of VAT (Total).

• To calculate the VAT on your sales: select cell E3 and enter the formula =F3*17.5%. With cell E3 still highlighted, press ^C or open Worksheet ➔ Edit and select Copy. Account books have about forty lines on a page - select cells E4 to E38 and press ^V or open Worksheet ➔ Edit and select Paste.

• To calculate total sales, select cell F39 and enter the formula =SUM(F3:F38). To calculate total VAT, with cell F39 still highlighted, press ^C or open Worksheet ➔ Edit and select Copy, select cell E39 and press ^V or open Worksheet ➔ Edit and select Paste.

• To calculate sales inclusive of VAT, select cell D3 and enter the formula =E3+F3. Copy this formula to cells D4 to D38. Copy the formula in cell E39 to cell D39.

• Select cells D3 to F39; move the pointer up to the Control Bar, click on the arrow to open the Style Menu, and click on currency.



The payments part of the accounts is more complicated because, in addition to the total cost of any purchase, you need to record the VAT and net purchase price separately.

• To calculate the VAT component of any total payment you enter, select cell L3 and enter the formula,

= 0.175/1.175*K3

Copy this formula to cells L4 to L38

• To calculate the total VAT, select cell L39

and enter

= SUM(L3:L38)

• For simplicity, this example assumes that all your purchases fall into one of three types: 'materials', 'motoring' or 'phone' (columns M,N,O). Ideally, you want to be able to enter a value into the total column and leave Eureka to calculate the VAT and the net price and insert the latter into the correct column.

This is the purpose of the 'type' column I. By inserting a number into this column (1 for materials, 2 for motoring or 3 for phone) when you enter a payment total, you can indicate to Eureka what type of purchase you have made.

• Select cell M3 and enter the formula

= IF($I3=1,($K3-$L3),)

This tells Eureka to look in cell I3 and, if it contains a 1, calculate the net price (K3-L3) and display it in M3; but if I3 contains any other value to leave the cell empty. (Note that if Eureka is set to display zero values, a £0.00 will appear in an empty cell. To switch these off, open Worksheet Options → Window and deselect Zero values, click OK.)

• Copy the formula in cell M3 to cells M4 to M38.

To total this column, select cell M39 and enter the formula

=SUM(M3:M38)

• Enter the following formula into cell N3

= IF($I3 = 2,($K3-$L3),)

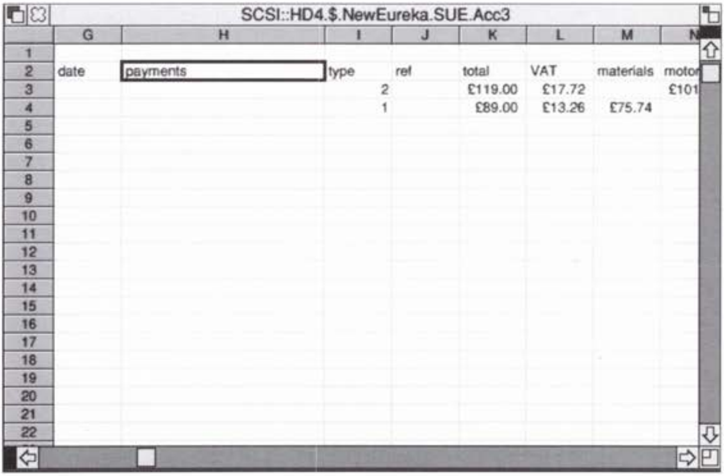
Copy it to cells N4 to N38. Total that column.

• Enter the following formula into cell O3

= IF($I3 = 3,($K3-$L3),)

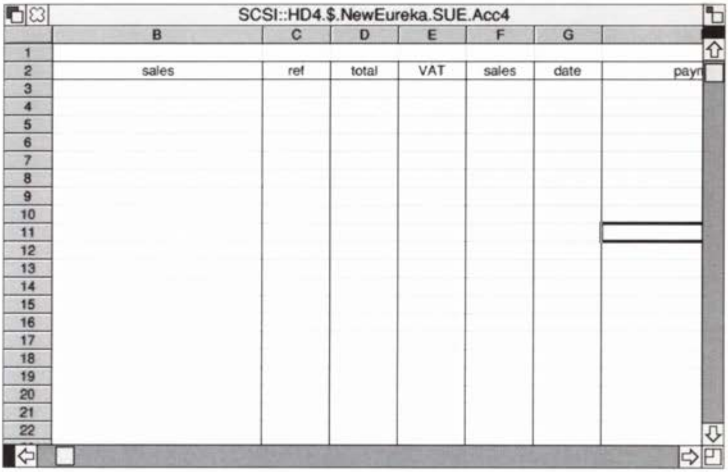
into cell O3, copy it to cells O4 to O38 and total that column.

• Select cells K3 to O39, and select the currency format from the style menu.



To format the accounts,
• Select cells A2 to S39. Press ^SHIFT F1 or open Worksheet > Format and select Border. Select Left and Right. Click OK.

Select cells A2 to S2. Click on the align centre button on the Control Bar to centre the labels in the columns. With these cells still selected, open the



Border dialogue, and select Left, Right, Top and Bottom. Click OK. Do the same with cells A39 to S39.

• To draw a thicker line between columns F and G and clearly divide the incoming and outgoing parts of the accounts, select cells G2 to G39, open the Border dialogue and click on left to erase the existing thin line. Select cells F2 to F39, open the Border dialogue, select a thick line style, then select right.

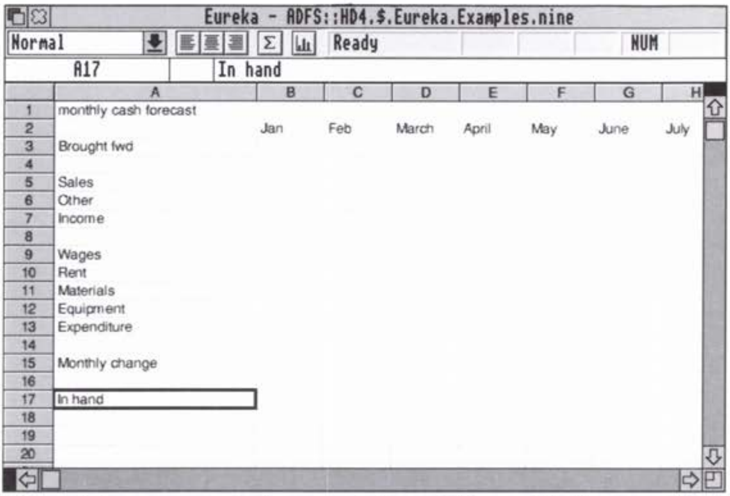
Cash Forecast

• Open a new worksheet.

• Column A will be used for labels. To widen it, position the cursor on the label bar, on the line between column A and column B. The pointer will turn into a double headed arrow. Press Select and use the mouse to drag the gridline until it lies over the line between columns C and D, release Select.

• Type in the labels as shown in the diagram: select each cell in turn, type the text into the data entry bar, then press Return.

• If you make a mistake: re-select the cell, press F2, and use the backspace key to correct the text.



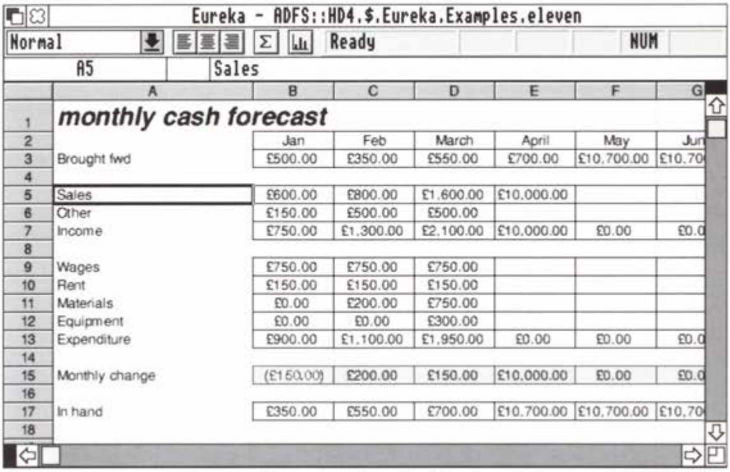
• From February onwards, the amount 'brought forward' is the amount 'In hand' from the previous month. Select cell C3 and enter =B17. Select cells C3 to M3, press ^F or open Worksheet ➔ Edit and select Fill; select Right from the Fill dialogue; then click OK.

• Calculate total income by selecting cell B7 and entering the formula =SUM(B5:B6); calculate total expenditure by selecting cell B13 and entering =SUM(B9:B12); calculate monthly change by selecting cell B15 and entering =B7-B13; and calculate the amount in hand by selecting cell B17 and entering =B3+B15.

• Use Fill to copy these formulas across all the months.

• Select cells B3 to M17, open the Style Menu on the Control Bar and select currency.

• Note that when you enter your data, if the numbers are too long to fit in the columns, Eureka will display ######## instead. To increase column width, select the columns, press ^SHIFT F3 or open Worksheet ➔ Format ➔ Column width. The current width of the columns is displayed in 0-character widths. Alter the number on the spaces between the columns and click on OK. Alternatively, you can double click the column border and Eureka will automatically re-size the column to fit the data.



	A	B	C	D	E	F	G
1	***monthly cash forecast***						
2		Jan	Feb	March	April	May	Jun
3	Brought fwd	£500.00	£350.00	£550.00	£700.00	£10,700.00	£10,70
4							
5	Sales	£600.00	£800.00	£1,600.00	£10,000.00		
6	Other	£150.00	£500.00	£500.00			
7	Income	£750.00	£1,300.00	£2,100.00	£10,000.00	£0.00	£0.0
8							
9	Wages	£750.00	£750.00	£750.00			
10	Rent	£150.00	£150.00	£150.00			
11	Materials	£0.00	£200.00	£750.00			
12	Equipment	£0.00	£0.00	£300.00			
13	Expenditure	£900.00	£1,100.00	£1,950.00	£0.00	£0.00	£0.0
14							
15	Monthly change	(£150.00)	£200.00	£150.00	£10,000.00	£0.00	£0.0
16							
17	In hand	£350.00	£550.00	£700.00	£10,700.00	£10,700.00	£10,70
18							

• Format the rest of the worksheet, using the Border dialogue (^SHIFT F1 or Worksheet ➔ Format ➔ Border) to draw lines and columns. If you want to draw coloured borders, select the colour first, then choose linestyle and position. Remember that you can select a number of unconnected blocks, by holding the CTRL key whilst you make the selections, then apply the same format to all the blocks. You may also want to use the Background dialogue (^F4 or Worksheet ➔ Format / Background) to apply a background tint to some of the boxes to highlight important information.

If you make a mistake, you can undo the formatting (but leave data and formulas unaffected) by selecting the appropriate cells, opening Worksheet ➔ Edit ➔ Clear and selecting Formats, then clicking OK.

Quick Reference Guide

=	*indicates that what follows is a formula*
Absolute reference	(F4) *this is a cell address that will not be adjusted if the formula is copied to another row or column: position cursor on address and press F4 to insert necessary dollar signs*
Alignment	Worksheet➔Format➔Alignment (^F2) *aligns data within cells. Data can also be aligned using the alignment buttons on the Control Bar*
Background	Worksheet➔Format ➔ Background (^F4) *sets background colour of cells*
Area	Chart ➔ Gallery ➔ Area *converts chart to area chart*
Attach Text	Chart ➔ Options ➔ Attach Text *attaches text in data entry field to selected element of chart*
Axes	Chart ➔ Options ➔ Axes *controls display of chart axes*
Bar	Chart ➔ Gallery ➔ Bar *converts chart to bar chart*
Border	Worksheet➔Format ➔ Border (^SHIFT F1) *formats border, border colour, and line style of cells*
CALC	(appears in status bar) *indicates that data has been altered but manual calculation has not been performed* (F9)
Calculation	Worksheet➔Options➔Calculation *sets Eureka to manual or automatic calculation*

Chart	Eureka→New document→Chart (Chart button) *opens chart window and displays selected data in Chart Format*
CIRC	(appears in status bar) *indicates that worksheet includes a formula that refers to its own address*
Clear	Worksheet→Edit→Clear (^B) *deletes data from cells; three options: all data, formats, formulas only*
Column	Chart → Gallery → Column *converts chart to column chart*
Column width	Worksheet→Format→Column width (^SHIFT F3) *sets column width to specified size, standard or best fit; hides and unhides selected columns; columns can also be re-sized with the mouse: place pointer on label bar and drag gridline to new position*
Connected	Chart → Gallery → Connected *displays scatter graph with connecting line*
Copy	Worksheet→Edit → Copy (^C) *takes copy of data from selected cells for pasting elsewhere*
Cut	Worksheet→Edit → Cut (^X) *removes data from selected cells for pasting elsewhere*
Delete	Worksheet→Edit → Delete (Delete key) *deletes cells, rows or columns*
Edit mode	(SHIFT F2) *used when editing any text field*
EXT	(appears on status bar) *indicates Eureka is in keyboard mode*
Fill	Worksheet→Edit→Fill (^F) *fills specified cell or cells with data from selected cell*

Font	Worksheet➔Format ➔ Font (^F3) Chart ➔ Format ➔ Font *controls text formatting. Options include font, font style, point size, character width, colour, kerning, underlining*
Functions	Worksheet➔Formula➔Paste Function presents *over 150 common and specialised functions that may be used to construct formulas*
Gallery	Chart ➔ Gallery *presents range of chart styles, including Area, Bar, Column, Line, Pie and Scatter, plus stacked, percentage and connected options*
Goto	Worksheet➔Formula➔Goto (F5) *used to move quickly to a specified cell, without scrolling*
Gridlines	Chart ➔ Options ➔ Gridlines *controls display of background gridlines on charts*
Insert	Worksheet➔Edit ➔ Insert (Insert key) *inserts cells rows or columns*
Legend	Chart ➔ Options ➔ Legend *displays automatically compiled legend or key to chart data*
Line	Chart ➔ Gallery ➔ Line *converts chart to line graph*
Manual calculation	(F9) *forces calculation when Eureka is in manual calculation mode*
NUM	(appears in status bar) *indicates that number lock is on*
Number	Worksheet➔Format➔Number (^F1) Chart ➔ Format ➔ Number *controls numerical formatting*

Page setup	Worksheet➔File➔Page setup (SHIFT Print) *controls page layout options for printing, including margin sizes, paper orientation, headers footers, display of gridlines and row and column headings*
Paste	Worksheet➔Edit ➔ Paste (^V) *pastes cut or copied data into highlighted cell or cells*
Paste Function	Worksheet➔Formula➔Paste Function *pastes selected function into data entry field*
Paste Special	Worksheet➔Edit➔Paste Special (^A) *selectively pastes cut or copied data into an active cell or cells. Options include: all data, formulas only, values only and format only*
Pattern	Chart ➔ Format ➔ Pattern *controls style of various elements of a chart including axes, legend and borders*
Percentage	Chart ➔ Gallery ➔ Percentage *converts chart data to values between 0 and 1 for display as a percentage chart (axis labels should be set to percentage format)*
Pie	Chart ➔ Gallery ➔ Pie *converts chart to pie chart*
Print	Worksheet➔File➔Print Chart ➔ File ➔ Print *prints a document. Options include page selection and number of copies*
Protection	Worksheet➔Format➔Protection (^SHIFT F2) *prevents accidental modification of sheets*
Quit	Eureka ➔ Quit ✘ *exit Eureka*

Row height	Worksheet➔Format➔Row height (^SHIFT F4) *sets row height to specified size or standard; hides and unhides highlighted rows, rows can also be re-sized with the mouse: place pointer on label bar and drag gridline to new position*
Save	Worksheet➔File ➔ Save (SHIFT F3) Chart ➔ File ➔ Save *saves worksheet or chart*
Save as	Worksheet➔File➔Save as (F3) Chart ➔ File/Save *saves worksheet or chart with new name*
Scale	Chart ➔ Format ➔ Scale *controls scale of selected chart axis*
Scatter	Chart ➔ Gallery ➔ Scatter *converts chart to scatter graph*
SCRL	(appears on status bar) *indicates that scroll lock is on*
Stacked	Chart ➔ Gallery ➔ Stacked *on certain types of chart, presents different sets of data 'stacked' one on top of the other*
Text	Chart ➔ Format ➔ Text *controls background colour of text attached to chart elements*
Style	Worksheet➔Format ➔ Style (^F5) *creates formatting styles for addition to Style menu. Options include alignment, background fill, border style, font, number format, protection*
Style menu	(on control bar) *presents four predefined format styles (plus user defined styles) that may be applied to data in selected cells*

Window Worksheet→Options→Window
controls window display options, including gridlines, row and column headings and zero values

Worksheet Eureka→New document→Worksheet
opens new worksheet

Formulas

Introduction

As described in the tutorial, Eureka offers three basic cell data types: numeric, text, and formula. Formulas allow you to compute results based on the contents of other cells, and therein lies the real power of a spreadsheet program. When you modify the cells referenced by a formula, Eureka automatically updates the formula to reflect the new cell contents. This allows you to model various systems and easily change the parameters when:

- New data arrives.
- You want to see the effects of different values; for example, you might want to know what the profit margin would be if sales were to increase 25% in a given quarter, perhaps as the result of a proposed advertising plan, whose cost also factors into the profit margin.

Formulas can depend on cells and each other in arbitrarily complex ways, allowing you to create sophisticated models which calculate quickly and without the tedium of repetitive hand calculations.

In addition, Eureka's extensive formatting capabilities allow you to create attractive, effective presentations of your models, and its charting capabilities allow you to further analyse your data graphically.

Formulas in Eureka belong to one of two broad classes:

- Simple formulas, which return a single value.
- Array formulas, which return multiple values into a range of cells.

The following sections describe these two classes, as well as named formulas, which can simplify the formula-building process.

Simple formulas

Suppose you want to create a formula that returns the value of cell A1 plus one. The formula would be:

```
=A1+1
```

The equals sign introduces a formula in the formula bar; all formulas must begin with an equals sign. When evaluating this formula, Eureka examines cell A1; if it needs to be calculated, Eureka does so before using its value. This is called *natural order calculation*. It is "natural" because Eureka calculates *precedent* cells before using their values in other *dependent* cells. For example, if A1 referred to A2, Eureka would recalculate A2 if necessary before using its value in A1, and so on. After obtaining the value of A1, Eureka then adds 1 to it, storing the final value in the cell containing the formula.

Operators

The plus sign in the example above is called the plus *operator*. An operator acts upon one or more *operands,* which are values, cell references, names, etc. In addition, each operator has an associated *precedence,* a rule governing how Eureka evaluates a formula when different operators are used in the absence of parentheses. Eureka evaluates operators with higher precedence before operators with lower precedence. In the list below, operators are listed in order of decreasing precedence; that is, the operators with highest precedence are at the top of the list, while the ones with the lowest precedence occur at the bottom of the list. Operators with *equal* precedence are listed on the same line; Eureka evaluates operators with equal precedence from left to right.

Operator	Description
:	Range. (Applies to two references.)
Space	Intersection. (Applies to two references.)
,	Union. (Applies to two references.)
-	Negation. (Single operand.)
%	Percent. (Divides single operand by 100.)
^	Exponentiation. (x^y = x^y.)
* and /	Multiplication and division.
+ and -	Addition and subtraction.
&	Text concatenation or joining.
= <> < > <= >=	Logical comparison.

For example, consider the expressions below:

Expression	Description
A1:C5	Refers to the range defined by the two end-points, A1 and C5.
A1:B10 A5:C5	Notice the space between the two range references; this is the *intersection* operator, which returns a reference representing the cells its two reference operands have in common. In this case, ranges A1:B10 and A5:C5 share the range A5:B5, which is the result of this expression.
A1,B2,C5:D10	This specifies a multiple area; the *union* operator produces a reference including both its operands.

	To use this expression in a formula, you would have to parenthesise it, because formulas use commas to separate arguments, leading to an ambiguity. Explicit grouping with parentheses resolves this ambiguity in favour of the union operator.
-A1	Negates the value in cell A1; if A1 contains 2, the result is -2.
15%	Equals 0.15.
10^2	Equals 100.
"This is "&A1	If A1 contains "text.", the result is "This is text."
A1=A2	Equals TRUE if A1 equals A2, FALSE otherwise.
2*3-1	Equals 5. Note how the precedence rules come into play here. This formula could be interpreted as equivalent to (2*3)-1 or 2*(3-1). The higher precedence of the multiplication operator, *, selects the former, which is consistent with the rule, "perform multiplication and division before addition and subtraction".
2*3^2	Equals 18. Again, the precedence rules come into play, favouring 2*(3^2) over (2*3)^2.

Note that you can use parentheses to group expressions and override operator precedence; you may also use parentheses if you feel it makes the formula easier to understand, even if precedence correctly resolves any ambiguities, making parentheses strictly unnecessary.

Using functions

Eureka provides many functions, described in detail in the Function Reference. Functions simplify formulas and expand on the capabilities of the operators. For example, to add cells A1, A2, A3, A4, and A5, you could use the formula:

```
=A1+A2+A3+A4+A5
```

but it is much simpler to use the SUM function:

```
=SUM(A1:A5)
```

Not only is this simpler to type and understand, it behaves better when you insert and delete cells (see the Editing Worksheets chapter).

Eureka provides a comprehensive range of simple mathematical functions, including SUM, EXP, SQRT, and ABS, and it also provides sophisticated financial and statistical functions, as well as a complete set of text functions.

You can use functions in formulas much as you use values; that is, functions can be operands, allowing you to construct expressions such as:

```
=SUM(A1:A5)*SUM(B1:B5)
```

You can also *nest* functions; that is, use a function as an argument to another function, as in:

```
=EXP(LN(A1))
```

You can nest functions up to 7 levels deep.

Array formulas

Array formulas expand upon the simple formulas described above by allowing a formula to act upon and return multiple values. For example, suppose you have a column of values in A1:A5 and B1:B5, and you would like to multiply corresponding entries and sum the products. The simple formula would be:

```
=(A1*B1)+(A2*B2)+...+(A5*B5)
```

while the array formula would be:

```
{=SUM(A1:A5*B1:B5)}
```

The braces surrounding this formula signify that it is an array formula. Eureka provides them when displaying the formula in the formula bar; you should *not* supply them yourself. You lock in an array formula by pressing Ctrl-Shift-Ins, rather than Return, and if you have selected a range, Eureka fills the range with the array formula.

Obviously, the array formula above is much more concise than the simple formula; moreover, the array formula can easily be modified to accommodate 100-row ranges, if necessary, while the simple formula cannot be easily modified.

Before returning to this example, we need to describe the array data type.

Array constants

An array constant is a data item that contains multiple data items, which can be numbers, text, logical values, and error values, which comprise the other constant types. An array cannot contain cell references or names. Arrays are organised into rows and columns, much like the worksheet itself. For example, a 1 row by 3 column array containing numbers would be specified as:

`{1,2,3}`

while a 3 row by 1 column array containing numbers would be specified as:

`{1;2;3}`

Notice that entries in the same row are separated by commas, while entries in the same column are separated by semicolons. Thus, a 2 row by 3 column array containing numbers is specified by:

`{1,2,3;4,5,6}`

You can use *array constants* such as these in formulas, much as you use simple data types. That is, you can add a number to all entries in an array:

`{1,2,3}+1` equals

`{2,3,4}`

You can add two arrays:

`{1,2,3}+{2,3,4}` equals

`{3,5,7}`

When you add or multiply an array, for example, Eureka adds or multiplies corresponding members in the two arrays.

As mentioned previously, arrays can contain numbers, text, logical values, and error values, the other constant types. Thus, the following is a valid array:

`{1,"Text",TRUE;FALSE,#N/A,#INV!}`

Array Expansion

When you add a number to an array, multiply two arrays, or use arrays as function arguments, all arrays must have the same dimensions. If the arrays are of different sizes, Eureka expands the arrays as necessary so that they all are the same size. Eureka defines the common size as follows: the number of columns is taken from the array having the greatest number of columns, while the number of rows is taken from the array having the greatest number of rows. For example, if you attempt to add a 1 row by 3 column array to a 2 row by 2 column array, the arrays would be expanded to 2 rows by 3 columns. Note that in this context, a single datum is considered a 1 row by 1 column array.

Eureka follows these rules when expanding an array:

- If the array contains only a single row, and the expanded array is to contain multiple rows, Eureka duplicates the existing row in each of the new rows, all aligned with the first column. If the array is to become wider when expanded, Eureka fills in all other entries with the #N/A error value.

- If the array contains only a single column, and the expanded array is to contain multiple columns, Eureka duplicates the existing column in each of the new columns, all aligned with the first row. If the array is to become taller when expanded, Eureka fills in all other entries with the #N/A error value.
- If the array contains multiple rows and multiple columns, Eureka does not duplicate any entries in the expanded array. Rather, it simply fills in all other entries with #N/A error values.

The following table illustrates these rules:

Formula	Expands to	Yielding
={1,2}+1	{1,2}+{1,1}	{2,3}
={1,2,3}+{4,5}	{1,2,3}+{4,5,#N/A}	{5,7,#N/A}
={1,2;3,4}+{5,6}	{1,2;3,4}+{5,6;5,6}	{6,8;8,10}
={1,2,3;4,5,6;7,8,9}+ {1;2}	{1,2,3;4,5,6;7,8,9}+ {1,1,1;2,2,2; #N/A,#N/A,#N/A}	{2,3,4;6,7,8; #N/A,#N/A,#N/A}

Using Array Formulas

With the preceding as background, let's return to the example presented at the beginning of this section:

```
{=SUM(A1:A5*B1:B5)}
```

This array formula multiplies corresponding entries in the ranges A1:A5 and B1:B5 and sums the products. Breaking the formula down, we have:

```
A1:A5*B1:B5
```

which in an array formula, tells Eureka to:

1. Recalculate ranges A1:A5 and B1:B5 if necessary.
2. Load the values in A1:A5 into a temporary array and B1:B5 into a second temporary array (we will call them A and B, respectively).
3. Ensure A and B are the same size, expanding them if necessary. In this case, we have the two arrays:

   A = $\{a_1,a_2,a_3,a_4,a_5\}$

   B = $\{b_1,b_2,b_3,b_4,b_5\}$

   where the a_i and b_i represent the current values of cells A1:A5 and B1:B5, respectively.
4. At this point, Eureka has A and B, which are two temporary array

constants. Thus, it can find the array product AB by multiplying corresponding members, as described previously.

Now that Eureka has the array product AB, it can compute the SUM function. When SUM is passed an array argument, it adds up all members and returns a single number. So, internal to Eureka, the formula now looks like:

`SUM(AB),` or really

`SUM(`$\{a_1b_1, a_2b_2, a_3b_3, a_4b_4, a_5b_5\}$`),`

where the a_i and b_i represent the current values of cells A1:A5 and B1:B5, respectively. As you can see, using an array formula has allowed you to use the SUM function, while saving you the trouble of creating cells to hold the individual products.

The example above describes an array formula returning a single result. Eureka also supports array formulas returning multiple results. For example, suppose you have a worksheet in which A1:A5 contain numbers, and you want B1:B5 to hold the results of EXP(A1), EXP(A2), etc. You could enter =EXP(A1) in cell B1 and copy it through B2:B5, or you could enter the array formula {=EXP(A1:A5)} into the range B1:B5. You would do this by selecting B1:B5, entering "=EXP(A1:A5)" in the formula bar, and pressing Ctrl-Shift-Ins to Array-Enter the formula. You would then find the correct values in B1:B5. You will notice that Eureka has added braces to the formula in the formula bar, and as you move the cell cursor through the array range, the formula does not change. The formula remains the same because an array range is a single entity; all cells share the same formula, and in particular, you cannot modify an individual cell's value within an array range (although you *can* format the cells individually).

Note: because you cannot edit or insert or delete individual cells in an array range, you should use the multiple-result type of array formula only when the benefits outweigh this cost. In particular, not much is gained in the example above; however, if the formula had been somewhat longer, we would have realised some fairly significant memory savings, and calculation speed would also improve over the non-array formula approach. Other types of formulas, such as those using array functions, including MINVERSE, MMULT, and LINEST, should be entered in array ranges, if you want their results completely displayed.

Named formulas

Eureka allows you to name cell references, and more generally, arbitrary formulas. Using meaningful names can simplify formula creation, help make formulas more self-documenting, save memory, and make inter-document referencing or linking more reliable.

To define a name, perform the following steps:

1. Open the Define Name dialogue box by choosing Formula→Define Name or by pressing Ctrl-N.
2. Type the new name in the *Name* edit field. A name consists of 1-32 characters, and can contain upper and lower case alphabetic characters, digits, the underscore, and the full stop. A name *must* begin with an alphanumeric character or the underscore.
3. Type the formula into the *Formula* edit field, remembering to begin it with an equals sign. You can point to cells just as you can when editing in the formula bar.
4. Click *OK* or press Return to accept the new name definition and close the Define Name dialogue. If you have several names to add, click *Add* instead, as it does not dismiss the dialogue.
5. When you have finished adding names, click *Close* or the window close icon.

To delete an existing name, perform the following steps:

1. Open the Define Name dialogue by choosing Formula→Define Name or pressing Ctrl-N.
2. Select the name you wish to delete from the name list.
3. Click *Delete*.
4. When you have finished deleting names, click *Close* or the window close icon.

Once you have dismissed the Define Name dialog, Eureka recalculates any cells affected by the changes you made.

For example, if B1:B12 contains your monthly figures, you might find it useful to name B1:B12 "Monthly_Sales". You could then use the name in a formula such as:

```
=AVERAGE(Monthly_Sales)
```

As mentioned above, you can also name arbitrary formulas. For example,

`=8.5%,` and

`=AVERAGE(Monthly_Sales)`

are both valid name formulas.

Function error values

Functions in Eureka can return a variety of error values, listed in the table below. These values can also be used in formulas anywhere a number or other value can be used. A formula cell whose current result is an error value displays that error value centred in the cell.

Error value	Cause
#DIV/0!	Division by zero has occurred. Make sure that any cells acting as divisors are neither empty nor contain zero. If this error can also occur internal to a function, this is stated in the function description in the function reference.
#INV!	This error value is returned by functions whose arguments are invalid. For example, SQRT(-2) returns #INV!.
#MEM!	The system does not have enough free memory for Eureka to evaluate the formula. This error condition can arise when dealing with large temporary arrays, for example.
#N/A	No available value. The "lookup" functions can return this error code value, and it can result from array expansion.
#NAME?	The formula refers to an undefined or deleted name. You should check the spelling of names in the formula and compare them to the names in the Define Name dialogue box.
#NULL!	This error is the result of an empty intersection between two references joined by the intersection (space) operator.
#NUM!	Functions generally return this error value to indicate overflow or a bad intermediate result, which causes the function to fail. This value is also returned upon arithmetic overflow; for example, 1E200*1E200 equals #NUM!.
#REF!	Signifies an illegal reference was made. This error value can occur due to pasting over or deleting worksheet cells, when those cells are referred to elsewhere in the worksheet.

#VALUE!	This error value results when you use a value whose type is incorrect in the given context, and which Eureka is unable to convert to the proper type via its automatic type conversion (e.g. "abc"+2 results in #VALUE!).

Editing worksheets

Entering data in cells

Basic procedure

To enter data into a cell, perform the following steps:

1. Select the cell.
2. Enter the new data. When you start typing, the contents of the formula bar are replaced with the new data, the Status Bar indicates that Eureka has entered Enter mode, and the Cancel and Accept icons appear to the left of the formula bar.
3. Click Accept to lock in the new entry, or press Return. Alternatively, you can press one of the arrow keys, Tab, Shift-Tab, Ctrl-Tab, and Ctrl-Shift-Tab to lock in the entry and move the cell cursor in the indicated direction.

   If you want to cancel the entry, click Cancel or press Esc. Eureka returns to Ready mode, and the formula bar reverts to its original contents.

Editing an existing cell

To edit an existing cell, perform the following steps:

1. Select the cell.
2. Click in the formula bar, or press F2. The Status Bar indicates that Eureka has entered Edit mode, and the Cancel and Accept icons appear to the left of the formula bar.
3. Make any necessary changes to the contents of the formula bar. Unlike Enter mode, the cell movement keys do not lock in the entry and move the cell cursor; rather, they move the caret in the formula bar.
4. Click Accept or press Return.

   If instead you want to cancel the entry, click Cancel or press Esc. Eureka returns to Ready mode, and the formula bar reverts to its original contents.

Entering data into multiple cells

To enter data into multiple cells, perform the following steps:

1. Select the range to receive the new data.
2. Edit an entry as described above, *except,* instead of locking the entry in with Return or another method, press Ctrl-Ins. The entry is copied throughout the selection as if you had first made the entry in a single cell, copied that cell, selected the destination area, and chosen Paste

Special, selecting the Formulas option. This is called a *Fill-Enter* operation.

While it is always good to think ahead and select the proper range before beginning a Fill-Enter operation, Eureka does allow you to *extend* a selection while editing in the formula bar. That is, while editing, you can click Adjust to extend the selection, and as long as you are not planning on Fill-Entering a multiple area, you can generally get by with this method.

Entering array formulas

To enter an array formula, perform the following steps:

1. Select the *single* range to receive the new data.
2. Edit an entry as described above, *except,* instead of locking the entry in with Return or another method, press Ctrl-Shift-Ins. The entry is entered into the range as an array formula. This is called an *Array-Enter* operation.

To signify an array formula, Eureka encloses it within braces in the formula bar. When you edit an existing array formula, Eureka removes the braces; you should not re-enter them, as Eureka will do this for you when you lock in the changes. Once you are finished editing an array formula, and you want to accept the changes, you *must* Array-Enter it; attempting other methods results in an error.

While it is always good to think ahead and select the proper range before beginning a new Array-Enter operation, Eureka does allow you to *extend* a selection while editing in the formula bar. That is, while editing, you can click Adjust to extend the selection.

Copying cells

Basic procedure

To copy cells from one place to another on the worksheet, perform the following steps:

1. Select the cell or range you wish to copy. This defines the source area.
2. Choose Edit→Copy or press Ctrl-C. The source area will be deselected and surrounded by the Copy marquee, a cycling dashed rectangle. The Status Bar will indicate that you are now in Copy mode.
3. Click on the cell you wish to be the upper left-hand corner of the destination area.
4. Choose Edit→Paste or press Ctrl-V. The source range is copied (pasted) to the destination area.

5. Repeat steps 3 and 4 if you wish to make additional copies, or press Esc to exit Copy mode.

Making multiple copies with a single paste

The procedure described above assumes you wish to make a single copy of the source area with each paste. However, suppose you have a column of formulas in the range A1:A10, and you want to copy this area to ranges B1:B10, C1:C10, and D1:D10. To accomplish this, you would modify step 3 above by selecting the range B1:D10. After choosing Paste, you would find the source range had been copied throughout the destination area.

In general, if the destination area is not a single cell, it should be exactly the same size as the source area, or its dimensions should be an exact multiple of the source area's dimensions. For example, if the source area is a 3 row by 4 column range, the destination area could be a single cell, another 3×4 range, a 6×8 range, a 9×12 range, and so on.

As a special case, if the destination area spans only one row or column (for example, A3:D3, C1:C10), Eureka will deduce how many copies to make as follows. If you select a single-row destination area, Eureka considers the number of columns in the source area, and if this number evenly divides the number of columns spanned by the destination area, the quotient gives the number of copies. In the example above, rather than selecting the entire destination area, B1:D10, you could have selected B1:D1 and obtained exactly the same results. Since B1:D1 is a 3 column, single-row range, and since the source area spanned a single column, Eureka knows to make 3 copies. Similarly, if the source area is instead a 2 column range, in order to make, say, 3 adjacent copies of it, you would need to specify a single-row range spanning 6 columns (2 columns × 3 copies). The procedure for column-oriented destination ranges is analogous.

Finally, note that Eureka does *not* require the destination range to be contiguous; as long as individual ranges in a multiple selection follow the rules given above, you can copy to non-adjacent areas. In any case, if these rules are broken, Eureka will display an error message to inform you the source and destination areas have different (incompatible) shapes.

Copying row and column formats

Sometimes it is useful to copy row or column formatting, including height and width information. You do this by selecting entire rows or columns for both the source and destination areas. When you choose Paste, you'll find the row or column formatting has been copied along with the cells.

Copying and Paste Special

Normally, copying cells duplicates formulas and formatting in the destination area. If instead you want to copy only the formatting, formulas, or current formula values, use the Paste Special command. The steps are the same as for copying, except that rather than choosing Edit→Paste, you choose Edit→Paste Special, or press Ctrl-A. The Paste Special dialogue appears. At that point, the steps are:

1. Select the *All* button if you want Paste Special to act exactly like Paste. Select *Formulas* if you want to copy only formulas. Select *Formats* if you want to copy only formats. Select *Values* if you want to copy only current cell values (excluding formulas).
2. Click the *OK* button.

Copying cells using the Fill command

If you have already entered data into cells in the same row or column, and you want to make adjacent copies of that partial row or column, then use Worksheet→Edit→Fill. For example, if you have entered data into A1:A10, then you can quickly copy this range to the adjacent columns by selecting, say, A1:E10 and choosing the Fill command. The steps are:

1. Select the range you wish to fill.
2. Choose Worksheet→Edit→Fill. The Fill dialogue appears.
3. If the data you wish to copy lies in the top row of the selection, select the *Down* button. If the data lies in the left-most column, select *Right*. If the data lies in the bottom row, select *Up*. If the data lies in the right-most row, select *Left*. If you want to copy the formatting, make sure the *Copy formats* button is selected.
4. Click the *OK* button.

Moving cells

To move cells from one place to another on the worksheet, perform the following steps:

1. Select the cell or range you wish to move. This defines the source area.
2. Choose Edit→Cut, or press Ctrl-X. The source area will be deselected and surrounded by the Cut marquee, a cycling dashed rectangle. The Status Bar will indicate that you are now in Cut mode.
3. Click on the cell you wish to be the upper left-hand corner of the destination area.
4. Choose Edit→Paste or press Ctrl-V. The source range is moved (pasted) to the destination area, the marquee disappears, and Eureka returns to

Ready mode. Paste Special is not available when moving cells.

Clearing cells

To clear cell contents, perform the following steps:

1. Select the cell or range you wish to clear. Multiple areas are permitted.
2. Choose Edit→Clear or press Ctrl-B. The Clear dialogue appears.
3. Select the *All* button if you want to clear both cell formulas and formats. Select *Formulas* if you want to clear formulas only; here, "formulas" include cell values for numeric and text cells. Select *Formats* if you want to clear formats only.
4. Click *OK*. Eureka clears the selection.

Inserting cells

To insert blank cells into the worksheet, perform the following steps:

1. Select the area you wish to clear. For example, if you want to insert an entire row of cells at row 2, shifting the cells in row 2 and below down one row, select row 2.
2. Choose Edit→Insert or press Insert. Eureka shifts the cells in rows 2 and below down one row, inserting a row of blank cells at row 2. In effect, Eureka moves the range A2:IV16384 to cell A3.

Beyond simply inserting entire rows and columns, Eureka allows you to insert several contiguous rows or columns at once, and it also allows you to insert partial rows and columns.

For example, if you wanted to insert 5 rows, rather than 1, in the example above, you would select rows 2-6, as a single selection, in Step 1.

However, if you want to confine the insert operation to an area which is not a whole row or column, perform the following steps:

1. Select the area you wish to clear. For example, if you want to insert 2 columns in column B, confined to rows 2-5, select the range B2:C5.
2. Choose Edit→Insert or press Insert. The Insert dialogue appears. Since you want to insert columns, make sure the "Shift cells right" button is selected. Since you want to insert partial columns, make sure the "Whole rows/columns" button is not selected.
3. Press the *OK* button. Eureka inserts 2 columns, confined to rows 2-5, shifting the cells in column B and to the right 2 columns right. In effect, Eureka moves the range B2:IV5 to cell D2.

As mentioned above, when inserting cells, Eureka essentially performs a *move* operation, relaxed somewhat since attempting to move B2:IV5 to cell D2 would normally result in an error, as cells in columns IU and IV cannot be moved 2 cells to the right—that would shift them off the worksheet. Insert relaxes this restriction; however, Insert *will* present an error box if any of the cells that would be shifted off the worksheet contain data. That is, if any of the cells in IU2:IV5 contained data in the example above, Insert would disallow that particular request.

Note: when you select whole rows or columns and choose Edit→Insert, Eureka does not present the Insert dialogue, as the selection makes your intent clear.

Deleting cells

To delete cells from the worksheet, perform the following steps:

1. Select the area you wish to delete. For example, if you want to delete an entire row of cells at row 2, shifting the cells in row 3 and below up one row, select row 2.
2. Choose Edit→Delete or press Delete. Eureka shifts the cells in rows 3 and below up one row. In effect, Eureka moves the range A3:IV16384 to cell A2., filling in row 16384 with blank cells.

Beyond simply deleting entire rows and columns, Eureka allows you to delete several contiguous rows or columns at once, and it also allows you to delete partial rows and columns.

For example, if you wanted to delete 5 rows, rather than 1, in the example above, you would select rows 2-6, as a single selection, in Step 1.

However, if you want to confine the Delete operation to an area which is not a whole row or column, perform the following steps:

1. Select the area you wish to delete. For example, if you want to delete 2 columns in column B, confined to rows 2-5, select the range B2:C5.
2. Choose Edit→Delete or press Delete. The Delete dialogue appears. Since you want to delete columns, make sure the "Shift cells left" button is selected. Since you want to delete partial columns, make sure the "Whole rows/columns" button is not selected.
3. Press the *OK* button. Eureka deletes 2 columns, confined to rows 2-5, shifting the cells in column D and to the right 2 columns left. In effect, Eureka moves the range D2:IV5 to cell B2.

Note: when you select whole rows or columns and choose Edit→Delete, Eureka does not present the Delete dialogue, as the selection makes your intent clear.

How copying and moving affect references

When you copy or move cells, Eureka adjusts references in formulas to reflect the new cell positions. For example, if you have a formula =A1+1 in cell B1, and you copy it to cell B2, it becomes =A2+1. This is an example of a *relative* cell reference; the reference is relative to the cell using it, and in this case, it means "the cell one cell to the left." Thus, when copying, Eureka adjusts the reference so it continues to refer to the cell one cell to the left. On the other hand, it is often necessary for cell references to remain constant when copying; these references are called *absolute*. An absolute reference is signified by the use of dollar signs. For example, if cell B1 contains =$A$1+1, when it is copied to B2, it remains =$A$1+1.

The major rules governing reference adjustment when *copying* cells are:

- Relative references are adjusted.
- Absolute references are not adjusted.
- References in cells outside the source and destination areas are not adjusted.

The major rules governing reference adjustment when *moving* cells are:

- Relative references are not adjusted, unless they refer to cells within the source area.
- Absolute references are not adjusted, unless they refer to cells within the source area.
- References in cells outside the source area are adjusted, *providing* they refer to cells inside the source area, so that they continue to refer to the same cells. If a reference is for a range, then the source area must encompass the entire range for the reference to be adjusted.
- References to cells inside the destination area are converted to #REF! error values, as cells pasted over are considered deleted from the worksheet. If a reference is for a range, then the entire range must be pasted over for this conversion to occur. In particular, pasting over a range corner does not destroy the range reference.

When adjusting references, Eureka may generate an invalid reference. For example, an adjusted relative reference may lie outside the worksheet boundaries. Also, Eureka requires ranges to be specified in the form "upper-left corner:lower-right corner," and adjustment of ranges using mixed references may result in a reversal of the two end-points, which is an error condition. Rather than generate incorrect references, Eureka converts all such invalid references to #REF! error values.

How inserting and deleting affect references

As mentioned above, the Insert and Delete commands are accomplished via *move* operations, and reference adjustment for Insert and Delete is also performed similarly to the *move* operation. However, an important difference concerns adjustment of range references. Recall that range references are not adjusted when moving cells, unless the range reference completely lies within the source area. When inserting and deleting, however, it is desirable to allow ranges to grow and contract, so that if you have a formula =SUM(A1:B10), and you insert a row at row 9, the formula becomes =SUM(A1:B11), reflecting the insertion of the new row. Thus, the rule for range references when inserting and deleting is modified as follows. If you are inserting or deleting rows, and the range reference completely intersects the source area along its horizontal dimension, the reference will be adjusted. Similarly, if you are inserting or deleting columns, and the range reference completely intersects the source area along its vertical dimension, the reference is adjusted. For example, consider the sheet below:

	A	B	C
1	1	6	=SUM(A1:B5)
2	2	7	
3	3	8	
4	4	9	
5	5	10	

If you insert a row at row 4, the worksheet and formula become:

	A	B	C
1	1	6	=SUM(A1:B6)
2	2	7	
3	3	8	
4			
5	4	9	
6	5	10	

The formula accounts for the inserted row, and will include any new data you enter into cells A4:B4 in its calculation. Suppose instead you delete rows 4 and 5. The worksheet becomes:

	A	B	C
1	1	6	=SUM(A1:B3)
2	2	7	
3	3	8	

As you can see, Eureka has adjusted the range reference to account for the deleted rows, so that data previously outside the original reference and excluded from the SUM function is not shifted into the area referenced by the SUM function.

Creating Data Series

You can create three kinds of data series in Eureka:

- Date - You can choose to increment the series in Day, Weekday, Month and Year units. If you start a date series with 01-Jan-93, and choose a step value of months, Eureka will create a series 01-Feb-93, 01-Mar-93, 01-Apr-93 and so on, entered into each new cell in the selection.
- Linear - You can choose a step value and a stop condition. To create a series that decreases, enter a negative number for the step value. For example, if you choose a step value of 5, for a linear series staring at 0, you will create 0,5,10,15,20,25 and so on.
- Growth - You can multiply values by a constant factor. If you start a growth series from 1, with a step value of 2, you would create the series 1,2,4,8,16,32,64,128,256,512 and so on.

To create a series

1. Enter the starting value you wish to use in the first cell in the series.
2. Select the cells in the row or column that you wish to fill with the series, ensuring the cell cursor is in the cell of the series starting value.
3. Choose Worksheet→ Data→ Series.
4. Choose the direction, right, left up or down - this should usually be automatically selected for you.
5. Choose the type of series you want to create, Date, Linear or Growth. If you choose date, select the Date unit you want to use.
6. Enter the step value increment or decrement, or growth factor you want to use.
7. Enter the stop value or condition you wish to use if you want Eureka to stop filling the selection at a certain value.
8. If the cell containing the start value has a format, select copy formats if you wish each cell in the series to use this format.
9. Choose OK.

Sorting Data

You can reorganise data on a worksheet alphabetically and numerically with the Worksheet→ Data→ Sort command. You can sort the rows and columns in any selection, and you can sort in ascending or descending order. You can specify up to three sort keys. A sort key specifies which column to use when sorting columns, or which row to use when sorting rows.

The ascending sort order used by Eureka is:

- Numbers - From the largest negative to the largest positive number.
- Text - Eureka ignores case unless the case sensitive option is on.
- Logical values - FALSE before TRUE.
- Error values - All error values are equal.
- Blanks - Always sorted last.

If you sort in descending order, this order is completely reversed, apart from blanks which are always sorted last.

To perform a sort

1. Select the range you want to sort.
2. Choose Worksheet→ Data→ Sort.
3. Choose sort by rows or columns.
4. Enter the reference of your sort key in the first key field.
5. To sort by more than one key, enter a reference in the second or third key fields.
6. Select ascending or descending sort order, for each key entered.
7. If you wish the sort to be case sensitive, or 'as displayed', select the appropriate option for each sort key entered.
8. If you wish to not disturb your cell shading and cell border formatting, select the 'preserve borders and shading information' option.
9. Choose OK.

You can sort by more than 3 keys by sorting two or more times. Remember that the last sort takes precedence, so always plan to sort with the most important keys last. The Sort - 'As displayed' option uses the currently displayed text as opposed to the underlying values, which are used in the normal case, which can be useful in some situations involving complex formatting. Selecting this option will degrade performance somewhat.

Formatting Worksheets

Number formatting

Eureka provides a powerful number formatting system based on templates, which describe how numbers should appear. The following worksheet demonstrates some of the possible formats:

	Format	Enter	Display
Cell default	General	10 -10 1.2345 text	10 -10 1.2345 text
Currency	£#,##0.00_);(£#,##0.00)	12 -15 0 text	£12.00 (£15.00) £0.00 text
Date	d-mmm-yy	33298 32154	1-Mar-91 12-Jan-88
Verbose date	dddd, mmm d, yyyy	33298	Friday, Mar 1, 1991
Telephone number	(0000) 000000	5555123456	(5555) 123456
Weights	0.00 "Kg"	8	8.00 Kg
Verbose currency	0.00 "Dollars"	100	100.00 Dollars
Trailing dots	@*.	Amount	Amount...................

A template has up to four parts, or segments, separated by semicolons, organised as follows:

format-if-positive; format-if-negative; format-if-zero; format-if-text.

Eureka applies the first segment to positive numbers, the second to negative numbers, the third to zero, and the fourth to text. If you omit the second and third segments, Eureka uses the *format-if-positive* segment to format negative and zero numbers; similarly, if you supply the second segment but omit the third segment, Eureka uses the first segment to format zero numbers. If you provide the *format-if-text* segment, Eureka uses it to format text entered in the cell; otherwise, Eureka displays the unformatted text.

Built-in formats

Eureka provides 21 built-in formats, which cannot be deleted or altered. To apply a built-in or custom format, perform the following steps:

1. Select the area you wish to format. This can be a single or multiple selection.
2. Choose Worksheet→Format→Number or press Ctrl-F1.
3. Select the format from the scrollable list.

4. Press OK. You can also double-click a format to select it and apply it to the selection.

The built-in formats are listed below, in the order they appear in the Format→Number dialogue box:

Built-in format	**Enter**	**Display**
General	1.75	1.75
0	1.75	2
0.00	1.755	1.80
#,##0	1250.25	1,250
#,##0.00	1250.25	1,250.25
£#,##0_);(£#,##0)	1250.25	£1,250
	-1250.25	(£1,250)
£#,##0_);[Red](£#,##0)	1250.25	£1,250
	-1250.25	(£1,250) (in red)
£#,##0.00_);(£#,##0.00)	1250.25	£1,250.25
	-1250.25	(£1,250.25)
£#,##0.00_);[Red](£#,##0.00)	1250.25	£1,250.25
	-1250.25	(£1,250.25) (in red)
0%	0.526	53%
0.00%	0.0125	1.25%
0.00E+00	1.2E15	1.20E+15
d/m/yy	33500	19/9/91
d-mmm-yy	33500	19-Sep-91
d-mmm	33500	19-Sep
mmm-yy	33500	Sep-91
h:mm AM/PM	0.75	6:00 PM
h:mm:ss AM/PM	0.75	6:00:00 PM
h:mm	0.75	18:00
h:mm:ss	0.75	18:00:00
d/m/yy h:mm	33500.75	19/9/91 18:00

Custom Number Formats

If the built-in formats do not meet your needs, you can create a custom number format that does. You build a format template using a variety of format codes. Among many other capabilities, these codes allow you to precisely indicate where digits should appear, how many decimal places should be displayed, and whether the number is a percentage. They also allow you to insert plain text into the formatted number.

Remember that a template is comprised of as many as four distinct segments, organised as follows:

format-if-positive; format-if-negative; format-if-zero; format-if-text.

If you want to hide a class of entries, enter the required semicolons, but leave that segment blank. For example, ";;;" hides all entries. The template "General;;" hides only negative numbers; positive numbers, zero, and text are all displayed in the General format.

To create a custom number format, perform the following steps:

1. Choose Worksheet→Format→Number or press Ctrl-F1.
2. Enter the desired format codes in the *Format* editable field.
3. Click *Add* to accept the format without applying it to the current selection, or click *OK* or press Return to accept the format, apply it to the current selection, and exit the Number dialogue. If you intend to press *OK*, be careful to select the correct range before invoking the Number dialogue.

To delete a custom number format, perform the following steps:

1. Choose Worksheet→Format→Number or press Ctrl-F1.
2. Select the format you wish to delete from the list. Built-in formats cannot be deleted.
3. Press *Delete*. The format will be deleted, and the Number dialogue will remain on screen. Any cells or styles using that format will revert to the General format.

Format codes

Eureka supports the following format codes:

Format code	**Description**
General	Specifies that numbers should be displayed with as many decimal places as will fit in the column. If the number is too large to fit, even when formatted with zero decimal places, it is formatted in scientific notation, with as many decimal places as will fit in the column. If it is still too large to fit, even when formatted with zero decimal places, Eureka fills the cell with hash signs.

0	Digit placeholder. A character is always displayed when a 0 is encountered in a format. If a 0 occurs to the left of the decimal point, and there is an integer digit in the number at that position, the digit is displayed; otherwise, a leading zero is added. Scanning right to left, once the front-most 0 is encountered, Eureka adds any remaining integer digits in the number. If 0 occurs to the right of the decimal point, Eureka rounds the number to as many decimal places as there are 0's. If there is a fractional digit in the number at the position of a 0, then the fractional digit is displayed. Otherwise, a trailing zero is added.
#	Digit placeholder. This format code is similar to 0, except that leading and trailing zeros are not displayed.
. (full stop)	Decimal point. This format code determines the position of the decimal point in the formatted number.
, (comma)	Thousands separator. When this code appears between 0 or # codes, it signifies that thousands are to be separated by commas. If this code is not flanked by 0 or # codes, it causes the number to be scaled by 1000. For example, the format "0," divides the number by 1000, and 12000 is displayed as "12". The format "0,,.00" divides by 1,000,000, so that 15,100,000.5 is displayed as "15.1".
%	Percent. Eureka multiplies the number by 100 and displays a % character.

E, E+, E-, e, e+, e-	Scientific notation. These codes cause Eureka to display the number in scientific notation, and they should be followed by 0's or #'s. The E+ and e+ variants display the sign of the exponent, even if it is positive, while the E, e, E-, and e- variants display the exponent sign only if the exponent is negative. For example, if the format is "0.0E00", 1.2×10^{15} would be displayed as 1.2E15, while if the format is "0.0E+00" the number would be displayed as "1.2E+15".
m, mm	Display the month of the serial number as a number in the range 1-12. The "m" code displays the month as a one or two digit number, while "mm" displays it as a two digit number with a leading zero, if necessary.
mmm, mmmm	Display the month of the serial number as text. The "mmm" code displays the month as an abbreviated month name, while "mmmm" displays the full month name.
d, dd	Display the day of the month of the serial number as a number in the range 1-31. The "d" code displays the day as a one or two digit number, while "dd" displays it as a two digit number with a leading zero, if necessary.
ddd, dddd	Display the day of the month of the serial number as text. The "ddd" code displays the day as an abbreviated weekday name, while "dddd" displays the full weekday name.
yy, yyyy	Display the year of the serial number as a number. The "yy" code displays the year as a two digit number in the range 0-99, while "yyyy" displays it as a four digit number.
h, hh	Display the hour of the serial number as a number in the range 0-23, or 1-12, if an "AM" or "PM" code is also present. The "h" code displays the hour as a one or two digit number, while "hh" displays it as a two digit number with a leading zero, if necessary.

m, mm	Display the minutes of the serial number as a number in the range 0-59. The "m" code displays the minutes as a one or two digit number, while "mm" displays it as a two digit number with a leading zero, if necessary. In order for these codes to be interpreted as minute codes, rather than month codes, they must follow an "h" or "hh" code.
s, ss	Display the seconds of the serial number as a number in the range 0-59. The "s" code displays the seconds as a one or two digit number, while "ss" displays it as a two digit number with a leading zero, if necessary.
AM/PM, A/P, am/pm, a/p	Specify that Eureka should display times in AM/PM format, rather than 24-hour format. Eureka displays the AM/PM indicator at the position of the format code, using the capitalisation and long or short form as specified by the code.
_ (underscore)	Moves horizontally by the width of the following character before displaying the next character in the formatted number. This code is useful, for example, in the currency formats, when using proportional fonts. Negative numbers use a format such as "(£0,000.00)". while positive numbers use a format such as "£0,000.00_)". Although digits in most proportional fonts are the same width, the space and parentheses characters may not be. The "_)" sequence in the positive number format causes right-aligned positive numbers to be offset to the left by the width of a right parenthesis, and this results in proper vertical alignment of the decimal points in a column of numbers. Note: this format code can only be applied when Eureka is actually drawing cells. Thus, it is ignored when used in worksheet functions such as TEXT, which construct a formatted number as the function result.

*	Repeats the next character enough times to fill the column width. You can have only one * in a segment of a format. Note: this format code can only be applied when Eureka is actually drawing cells. Thus, it is ignored when used in worksheet functions such as TEXT, which construct a formatted number as the function result.
@	Specifies the position of text, if the item to be formatted is text rather than a number. You can have only one @ code in a format, and it must occur in the last segment, which becomes the "format-if-text" segment. If you omit the @ code from a format, and the format is applied to text, then the text is displayed without any formatting.
£ $ + - / ( ) : space	Display that character. To display other characters, escape them with the backslash or double quote, as described below.
\	Escapes the following character. Use the backslash character to remove the special meaning of a format code. Any character preceded by a backslash, including the backslash, is added to the formatted number without change.
"*text*"	Text appearing in double quotes is added to the formatted number unchanged. Use the double quote in preference to the backslash, when you have a sequence of several characters you want to include unchanged in the formatted number. To include a double quote character inside text enclosed by double quotes, escape it with a backslash.
[BLACK]	Display the formatted text in black.
[BLUE]	Display the formatted text in blue.
[CYAN]	Display the formatted text in cyan.
[GREEN]	Display the formatted text in green.
[MAGENTA]	Display the formatted text in magenta.
[RED]	Display the formatted text in red.
[WHITE]	Display the formatted text in white.
[YELLOW]	Display the formatted text in yellow.

[COLOUR *n*]	Display the formatted text in colour number *n* in Eureka's internal palette, where *n* is a number between 1 and 16. Note: the colour codes can only be applied when Eureka is actually drawing cells. Thus, they are ignored when used in worksheet functions such as TEXT, which construct a formatted number as the function result.

Aligning cell contents

Eureka provides several options for aligning cell data for display. The default cell alignment is called *General*.

When referring to horizontal cell alignment, General has the following effect:

- Numbers are aligned with right cell borders.
- Text is aligned with left cell borders.
- Logical values are centred in cells.

When referring to vertical cell alignment, General has the following effect:

- All data types are aligned with cell bottoms, unless the "Wrap Text" option is in effect.
- If the "Wrap Text" option is in effect, cell contents are aligned with the tops of cells.

If the contents of a text-valued cell are too long to completely fit in the cell's column, Eureka allows the text to spill over into neighbouring cells, providing those cells are empty. However, Eureka does *not* allow numeric cells to overflow a cell's left or right borders. Rather, Eureka fills the cell with as many hash # signs as will fit in the cell's column.

To change cell alignment, perform the following steps:

1. Select the range or ranges you want to alter.
2. Choose Format→Alignment or press Ctrl-F2. The Alignment dialogue appears.
3. Select the desired horizontal and vertical alignment options from the dialogue. If you want Eureka to word-wrap the text in the cell, select the *Wrap Text* button.
4. Click the *OK* button or press Return.

When wrapping text in a cell, and the row containing the cell has the default row height, Eureka increases and decreases the row height as necessary to accommodate the text. However, this height adjustment occurs only when entering data or changing the alignment; in particular, it does *not* occur during recalculation.

Fonts

Eureka supports use of as many as 256 fonts in a single worksheet. In this context, a *font* is a particular typeface and size, along with colour information and kerning and underlining status.

To alter the font used for display, perform the following steps:

1. Select the range or ranges you want to alter.
2. Choose Format→Font or press Ctrl-F3. The Font dialogue appears. The *Size* option allows you to specify the font size in points, where one point equals 1/72". The *Automatic Kerning* option, if selected, takes effect when used with the RISC OS 3 outline font manager, and fonts containing kerning information.
3. Make the desired modifications.
4. Click *OK* or press Return.

If speed is of the absolute essence, or you are working in a low resolution screen mode, you may want to use the RISC OS System font, the font you see in Filer windows and in the Icon Bar. This font cannot be scaled; it always has a height of 12 points, which is the size you should select.

The colour *Automatic* as used here results in black text.

Shading cells

To change the background cell colour, perform the following steps:

1. Select the range or ranges you want to alter.
2. Choose Format→Background or press Ctrl-F4. The Cell Background dialogue appears.
3. Select the colour you wish for the selection. The displayed colours are taken from Eureka's internal palette.
4. Click *OK* or press Return.

The colour *Automatic* as used here results in a transparent (white) background.

Cell borders

Eureka allows you to construct grids and tables and otherwise emphasise cells through the Format-Border command. You use the Format-Border command to select a line pattern, colour, and weight to apply to any of the four sides of a cell.

To change cell borders, perform the following steps:

1. Select the range or ranges you want to alter.
2. Choose Format-Border or press Ctrl-Shift-F1. The Cell Border dialogue appears.
3. Select the border colour and style from the *Style* area. Then click on the boxes indicating the cell sides to which you want to apply or remove a border. If you want to surround a selection with a border, click the *Outline* box.
4. Click *OK* or press Return.

The colour *Automatic* as used here results in black borders.

Protecting cells

Eureka offers two levels of security for protecting cell contents from modification and inspection.

A *locked* cell cannot be changed, provided the worksheet is protected; attempts to modify it directly, format it, move it, paste over it, or shift its location via the Insert and Delete commands are denied. You can, however, copy a locked cell to another cell. By default, all cells on a new worksheet start out locked; however, the locked attribute only takes effect when worksheet protection is enabled via Options-Protect Document. Thus, the normal state allows editing of cells. Once you have enabled worksheet protection, you will not be able to do much in the way of modifying a locked cell, so if you intend for the user to be able to alter the cell, you should unlock it prior to protecting the worksheet.

To lock or unlock a cell, perform the following steps:

1. Select the range or ranges you wish to modify.
2. Choose Format-Protection or press Ctrl-Shift-F2. The Cell Protection dialogue appears.
3. Select or deselect the *Locked* button as desired.
4. Click *OK*.

Eureka also permits hiding cells to prevent their formulas from appearing in the formula bar. To hide or unhide a cell, perform the following steps:

1. Select the range or ranges you wish to modify.
2. Choose Format→Protection. The Cell Protection dialogue appears.
3. Select or deselect the *Hidden* button as desired.
4. Click *OK*.

As with the *locked* attribute, the *hidden* attribute only takes effect when worksheet protection is enabled.

Note: if you want to prevent the cell from being displayed in the worksheet itself, create a custom number format such as ";" and apply it to the cell. See the section on Number Formatting for more.

Lotus 123 Files

Eureka can load and save Lotus 123 WKS and WK1 files. While worksheet data and most formatting is preserved, graphs and Lotus 123 features which Eureka does not directly support are not preserved.

Loading Lotus 123 Files

To load a Lotus 123 file, drag it to the Eureka Control Panel or the Eureka icon on the Icon Bar. Make sure the file has the RISC OS file-type "Lotus123".

When Eureka loads a Lotus 123 file, it will open a single window to display it. The upper left cell in the window and cursor position will agree with the values stored in the Lotus 123 file, but title rows, window splits, and the Lotus 123 window size are ignored. Column widths and hidden columns are properly translated.

In general, when loading a Lotus 123 file, Eureka does a good job of emulating Lotus 123 cell formatting. However, the Lotus 123 "+/-" and "Text" formats are converted to General, as they have no counterpart in Eureka. Eureka creates custom number formats to provide for Lotus 123 formats like "Fixed, with 8 decimals." The Lotus 123 hidden format is translated to the Eureka format ";;;". Eureka translates the Lotus 123 international date and time formats to "d-mmm-yy" and "h:mm:ss". Eureka modifies the Normal style to agree, as closely as possible, with the default format indicated in the Lotus 123 file. However, Eureka ignores the "global label alignment" feature supported by Lotus 123, as Eureka has no such concept.

Saving Lotus 123 Files

When saving a file in Lotus 123 format, almost all the built-in Eureka formats are fully translated. In addition, Eureka is able to translate simple custom formats, such as "0.0000" and "0.0000E+00", to Lotus 123 equivalents. Eureka translates formats it cannot convert to the Lotus 123 General format.

Formula Translation

All Lotus 123 functions have Eureka counterparts and translate. However, Eureka defines some operators and functions differently than Lotus 123.

- The exponentiation and negation or unary minus operators have different precedence in Lotus 123 and Eureka. For example, in Eureka, -4^2 equals 16, while Lotus 123 calculates -16. Use parentheses to correct this problem.

- Eureka's % operator has no Lotus 123 counterpart, but Eureka translates it by dividing its operand by 100.
- The Lotus 123 unary plus operator is ignored when reading Lotus 123 files, as it appears to be ignored by Lotus 123 during calculation, but has a subtle effect in Eureka, where it coerces its operand to a value.
- Eureka's INT function returns the smallest integer not greater than its argument, while Lotus 123's INT functions truncates its argument, so INT(-2.2) equals -3 in Eureka and -2 in Lotus 123. The MOD functions in the two programs are each consistent with their complementary INT functions, but they also differ in how they deal with negative arguments.
- In Eureka, the VLOOKUP, HLOOKUP, INDEX, and CHOOSE functions all index their arguments beginning with 1, while in Lotus 123, indexes begin with 0. Eureka adds or subtracts 1 from the index arguments as necessary,
- Eureka's financial functions are more powerful and offer additional options compared to Lotus 123's; also, Lotus 123 does not generally follow the convention that cash outflows are negative while inflows are positive. Eureka does whatever is necessary to ensure that the result of the function agrees with the result in the host program (this involves changing the sign of one of the arguments, reordering arguments, and sometimes inserting a missing argument).
- There are a number of other subtle differences between functions. For example, Lotus 123's @AVG counts label cells, while Eureka's AVERAGE ignores them. Lotus 123's sort order can differ from Eureka's, so functions like VLOOKUP may not be portable between the programs if the lookup range contains data types differing from the type of the search value.
- Arrays, union and intersection operators, and other Eureka-specific features don't translate to Lotus 123 files.
- Eureka translates @NA to #N/A and @ERR to #VALUE!. Except for #N/A, all Eureka error codes map to @ERR.

- Some Eureka functions, although supported by Lotus 123, do not translate because more arguments were specified than are used in Lotus 123, or the arguments given are not compatible with Lotus 123's functions. For example, the Eureka financial functions are based around a more flexible equation than Lotus 123's. If you specify a *type* of 1 in PV, this causes payments to be due at the beginnings of periods, rather than at the end, and this is incompatible with Lotus 123.
- The text functions FIND, MID, and REPLACE also suffer from off-by-one arguments, similar to VLOOKUP et. al. discussed above, and these text functions are also corrected.
- Eureka supports missing arguments in formulas, while Lotus 123 does not. Eureka replaces a missing argument with its default value when writing a Lotus 123 file. For text arguments, this is always the empty text, "". For numbers it varies; for example, IF(1,2) becomes @IF(1,2,0), while IRR(A1:A5) becomes @IRR(0.1,A1:A5).

Whenever a cell fails to translate, Eureka opens an alert box noting the reference of the cell, and offering a chance to cancel the operation completely, or to continue, either reporting subsequent errors or not. Cells containing formulas which fail to translate are simply replaced with their current values.

Eureka$LotusFont

This environment variable is analogous to Eureka$WorksheetFont. Eureka$LotusFont controls the font used in the Normal style when loading a Lotus 123 file. Because Lotus 123 typically uses fixed-pitch fonts, you will probably also want to use fixed-pitch fonts, such as System and Corpus. If this variable is undefined, Eureka uses the RISC OS System font.

For example, the following line in Eureka's !Run file causes Eureka to use Corpus.Medium, 10 point, in Lotus 123 files it imports.

Set Eureka$LotusFont : +font=Corpus.Medium +size=10

Linking Worksheets

Eureka allows you to refer to other worksheets by including their names within the cell references. For instance, if worksheet Sheet1 wants the value of cell A1 in Sheet2, this can be obtained with the reference Sheet2!A1. The '!' character indicates the link.

Worksheets containing links to other files are said to be *dependent* on them; those other files are the *supporting* worksheets.

While a supporting worksheet is open, the worksheet name alone is enough to identify the file to link to. If however a supporting worksheet is *not* open, more information is needed: specifically, where the supporting worksheet is stored relative to the dependent worksheet.

Consider an open dependent worksheet Sheet1, and its closed supporting worksheet Sheet2.

If Sheet1 and Sheet2 are in the same directory, a reference to Sheet2 will appear as Sheet2!ref. If Sheet2 is one directory down from Sheet1 (say in 'Subdir'), the reference will appear as Subdir.Sheet2!ref. If Sheet2 is *up* one directory from Sheet1, it will appear as '^.Sheet2'!ref.

A full pathname is used if the external reference lies on a different filing system, or if the dependent worksheet has not yet been saved; for instance 'adfs::4.$.Sheet2'!ref.

Note that complex filenames involving characters such as '^' and ':' are contained in single quotes, so that Eureka does not try to interpret them as part of the formula.

When open, worksheet names must be unique to allow unambiguous linking. If Eureka detects a duplicate name, you are given the option to assign it a distinct name, such as Sheet3.

Calculation between worksheets can proceed only when supporting worksheets are open. Until then, cells dependent on unopened worksheets show the value #REF!. When you open a dependent worksheet, Eureka offers to open any supporting worksheets not already open.

If changes are made to a supporting worksheet, only those dependents currently open will recalculate; unopened dependents are not affected. In particular, if you save a supporting worksheet under a new name, only dependent worksheets that are open change to reflect the new name. This may cause problems when the unaltered dependents are opened, as they will not be dependent on the correct worksheet. The solution in this case is to use the Change Links dialogue to redirect the link to the correct file.

Changing Links

To change the links of a dependent worksheet, perform the following steps:

1. Open the dependent worksheet, and ensure it is active if other worksheets are already open.
2. Choose Worksheet→File→Change Links. The Change Links dialogue appears, displaying a list of supporting documents for this worksheet.
3. Select the link you wish to change. You can deselect selected items using Adjust or shift-clicking.
4. You can now edit or type the new filename into the edit field, use the drop-down box to pick the new filename, or drag a file into the dialogue box.
5. Click *OK* to accept the change and exit the dialogue, or *Change* to accept the change but keep the dialogue on screen. If you make no selection and the edit field is empty, you can exit the dialogue without change by clicking *OK*.

After you dismiss the dialogue, Eureka will recalculate the worksheet.

How Copying and Moving Affect External References

When you copy or move cells within a worksheet, cell references are altered according to simple rules; please see that section for the details.

Eureka consistently applies those rules for external references also. When creating linked documents, it may be worth bearing the following example in mind:

Sheet 1

	A	B
1	99	66
2	=A1	
3	=Sheet1!A1	

Sheet 2

	A	B
1	42	79
2		
3		

Sheet1's A2 and A3 will of course both display the value 99.

Now copy the range A2:A3 from Sheet1 to Sheet2, pasting it to the range A2:B3. Sheet2 displays:

Sheet 2

	A	B
1	42	79
2	42	79
3	99	66

Sheet2's A2 and B2 contain "=A1" and "=B1", as if they had been copied within Sheet1 — but they now refer to Sheet2's cells.

Sheet2's A3 and B3 contain "=Sheet1!A1" and "=Sheet1!B1", thus referring to Sheet1's cells as if copied within that worksheet.

Note that names are pasted along with formulas if necessary.

Name Conflicts

If a pasted name conflicts with an existing one, a dialogue box appears letting the user accept or ignore the name or cancel the operation entirely.

Charting

You create and display a chart by selecting a range and then either clicking the chart icon or choosing Eureka→New Document→Chart. When this happens, a link is made from the chart to the worksheet containing the data, in the same way that links can be made between worksheets. This means that when the data in the worksheet changes, the chart updates automatically. As a dependent document, charts do not alter unless supporting worksheets are open; unlike worksheets (which display #REF!) they will continue to show the old values. In other respects charts behave as dependent documents.

Consequently, the same Change Links dialogue is available and works as for dependent worksheets.

When you create a chart, Eureka examines the composition of the selected range to determine in what form the chart should appear.

Eureka assumes there will be fewer series than categories; so for example, given a worksheet like this:

	A	B	C	D	E
1		Jan	Feb	Mar	Apr
2	Celery	2	4	4	6
3	Apples	4	3	2	5
4	Walnuts	1	5	6	5
5	Grapes	4	4	6	4

the following cases arise:

1. Creating a chart for the selection A1:C5 (more rows than columns) produces a column chart with four categories (Celery, Apples, Walnuts and Grapes) and two series (ie. two bars per category) for Jan and Feb. The same applies for selections B1:C5 (missing category labels) and A2:C5 (missing series labels).
2. Charting A1:E3 (more columns than rows) produces a column chart with four categories (Jan, Feb, Mar and Apr) and two series for Celery and Apples. The same applies for A2:E3 (missing category labels) and B1:E3 (missing series labels).
3. If only one row or column is selected, with or without the labels, Eureka produces a column chart with one series.

4. If no obvious labels are present and more than one row or column is selected, for instance the selections B2:C5 or B2:E3, Eureka assumes you want a scatter chart. If there are more columns than rows, X values are taken from the first row in the selection. If there are more rows than columns, X values come from the first column.
5. Selecting A1:E5 or B2:E5 (equal columns and rows, with or without labels) produces a conflict; Eureka cannot choose between the alternatives, so the New Chart dialogue appears.

The New Chart Dialogue

This dialogue appears if Eureka cannot decide in which form your chart should be displayed. It can also be made to appear by shift-clicking either the chart icon or the Eureka→New Document→Chart menu item, or clicking Adjust on the chart icon — this lets you override Eureka's choice of chart.

The dialogue allows you to decide whether the series are laid out in rows or columns, and whether the first of these rows or columns contain category labels, the first series, or X values for a scatter chart.

Chart Types

There are six types of chart. By default, a column or scatter chart is chosen as described above. To switch to another type, open Chart→Gallery and select from the list.

Area, Bar, Column and Line charts can optionally be displayed stacked — the series values are shown stacked upon one another to give a cumulative total; or as a percentage — each series is shown as a percentage of the total of all series for that category. In the latter case the value axis ranges from 0 to 1; changing the number format for the scale values will allow them to be displayed as percentages.

Scatter charts can also be shown with the data points connected.

Pie charts are based on the first series only.

Chart Elements

These are the constituent parts of the chart display. In the same way that worksheet cells have formats associated with them, so do chart elements. By altering these formats variations on the standard chart can be produced.

When first displayed, a chart has these elements:

- Chart Background: equivalent to the paper the chart is drawn upon;

- Category Axis: displaying the category labels, if these were selected along with the chart data;
- Value Axis: displaying the range of values that encompasses all the data;
- Plot Area: the data display area, containing one or more series;
- Legend: displaying a key identifying each series, if more than one are present and if series names were selected along with the chart data;
- Series elements: there is one element for each series in the data to be plotted.

Selecting Chart Elements

Chart elements can be selected by clicking on the relevant areas of the display; a selected element is displayed with handles surrounding it. (In the case of Series elements, the handles are shown on the first, middle and last data value of that series, or around the circumference of a pie chart.) Alternatively, the cursor keys may be used to cycle through the elements.

Once selected, formats can be changed for that element by opening Chart→Format and selecting the required format. If a format change is not applicable for that element, Eureka will beep.

Series Elements and Functions

Each Series element corresponds to a series in the chart. It is associated with a SERIES function, displayed in the formula bar when the element is highlighted. It is this SERIES function which describes the data needed for the series in the chart.

SERIES functions are special, in that you are not allowed to enter them directly into worksheets; you can only edit them through use of charts. This is achieved either by editing in the formula bar, or by using the Chart→Edit→Series menu item.

Formula Bar Editing

Directly editing SERIES functions is perfectly allowable, but is not recommended for the beginner.

The arguments to the SERIES function are as follows:

1. The series name — where to find the name to place in the Legend if there is one.
2. The category names or values — where to find the labels for the category axis, or X values for a scatter chart. For labels, this argument is essential only for series 1.

3. The series values — where to find the values to place in each category for this series.
4. The plot order — a number determining when in the sequence of series this series is plotted. Lower numbers are plotted first. A series retains its colour whichever position it is plotted in.

Not all arguments need be present. Names are allowed. You must always specify the link to the worksheet containing the data. You can point to cells in your worksheet.

The second argument serves a dual purpose. If a scatter chart has been selected, then it determines where to find X values for that chart. It is possible, using this method, to overlay two scatter charts containing independent X values; an example is given later in this section.

The Edit Series Dialogue

As an alternative to editing in the formula bar, you can edit SERIES functions using the Edit Series dialogue.

To do this, perform the following steps:

1. Choose Chart→Edit→Series or press Ctrl-S. The Edit Series dialogue appears. A scrollable list contains the names of all the series in your chart. (If you did not include labels in the charted data, these names are "Series 1", "Series 2", etc.)
2. To add a new series to your chart, type in its details (described below), and click on *Add*. To delete a series, select it from the list and click on *Delete*.
3. When a series is selected, the details of the series are given in four edit fields so you can alter them. These fields are:
   - Name — where to find the name for this series, which is placed in the Legend if there is one.
   - Category — where to find the labels for the category axis, or X values for a scatter chart. For labels, an entry is essential only for series 1.
   - Value — where to find the values to place in each category for this series.
   - Plot order — a number determining when in the sequence of series this series is plotted. Lower numbers are plotted first. A series retains its colour whichever position it is plotted in.

Note that each entry is a formula. Not all entries need be present. Names are allowed. You must always specify the link to the worksheet containing the data. You can point to cells in your worksheet once you have inserted an "=" in the edit field.

The Category field serves a dual purpose. If a scatter chart has been selected, then it determines where to find X values for that chart. It is possible, using this method, to overlay two scatter charts containing independent X values; an example is given later in this section.

4. Make your changes to this series.
5. If this is the only series you wish to change, you can then click *OK* or press Return to keep the changes and leave the dialogue. If you want to alter another series, click on *Accept* to keep the changes and stay in the dialogue. Clicking on *Close* without clicking on *Accept* ignores any changes to the edit fields.

Two other buttons are present in the dialogue. The *Options* button takes you to the Series dialogue produced by Chart→Options→Series, so you can act on the currently-selected series (see Best-Fit Lines, below). The *Pattern* button will allow customisation of Series patterns (see Patterns, below).

Scatter Charts with Independent X Values

By editing SERIES functions either in the formula bar or by using the Edit Series dialogue, Eureka allows you to overlay scatter charts that have independent X values. An example is given below; note that the description explains the method using editing in the formula bar, as this is the less intuitive of the two methods.

Consider the following worksheet:

Sheet1

	A	B	C	D
1	X1	Y1	X2	Y2
2	0	2	1	4
3	3	4	2	5
4	4	5	3	6
5	6	3	2	8

We would like to produce a scatter chart with two connected paths on it, denoted by the coordinates (X1,Y1) and (X2,Y2) down the worksheet. We thus need two series (for each Y coordinate), but with independent X values.

To do this, perform the following steps:

1. Select the range A1:C5, and shift-click on the Chart icon. Specify that the first column contains X values for a scatter chart, and click *OK*. This gives us a scatter chart with two series; the column under X1 gives us the X values for both Y1 and X2. The first series is correct, although the second is not.
2. Select the second series by using the arrow keys until Series 2 appears next to the formula bar. We now need to alter the SERIES function, so press F2 or click in the formula bar.
3. The first three arguments need changing. The series name argument should point to Sheet1!D1 (for "Y2"), so either type that or point to the cell in the worksheet (the reference can be either absolute or relative). Similarly the category names/values argument should be changed to Sheet1!C2:C5 (to take X values from the column under X2), and the series values argument to Sheet1!D2:D5 (to take values from the column under Y2).
4. Click Accept or press Return to commit the function. Both series are now correct.
5. Change to a connected scatter chart using Chart→Gallery→Connected.

Editing SERIES functions in this fashion is perfectly allowable, and can achieve results that cannot be achieved in other ways. However, it is not recommended for the beginner.

Adding Text to Charts

Additional Chart Title and Axis Title elements can be created to label the chart and/or axes; to do this, highlight the element to be labelled and either start typing, or choose Chart→Options→Attach Text (a default title is created in this case) which has the short-cut Ctrl-T. Please note that rotated text for axis labelling is not supported for RISC OS 2 users. If you require this facility, you should upgrade to RISC OS 3.1.

Pie charts are limited to having only a Chart Title element.

Note that the Chart/Axis Title elements can be arbitrary formulas, introduced with '=' in the usual fashion.

Formatting Charts

Patterns

For Chart Background, Plot Area, Legend and Axis/Chart Title elements, the Pattern format dialogue allows you to define a border colour, style, line weight and shadow type, and also a background colour (fill) for that element.

For the Category Axis and Value Axis elements, the pattern dialogue allows you to choose the line style, weight and colour for the axis, and styles for the major and minor ticks along each axis. You can also alter the positioning of labels with respect to the axis line.

For Series elements, the pattern dialogue lets you customise the way the data is displayed according to the type of chart chosen. Lines, markers and colours can all be altered as appropriate.

To change the pattern of a chart element, perform the following steps:

1. Select the element you wish to alter, by clicking on it or using the cursor keys.
2. Choose Chart→Format→Pattern or press Ctrl-F4. The Pattern dialogue appears.
3. Select the pattern type according to the chart element chosen.
4. Click *OK* or press Return to exit, or click Menu to alter another format.

For Border and Axis settings, *Automatic* means a solid black line of weight 0.5; for Fill settings, a white background.

Ticks, the marks along the axes, may be outside, inside or cross the axis (or not there at all).

Labels may be adjacent, high (above/to the right of the axis), low (below/ to the left of the axis) or be turned off.

Font settings

For the Legend, Value/Category Axis and Axis/Chart Title elements, you may choose the font to be used for display.

To alter the font, perform the following steps:

1. Select the element containing the font you wish to alter.
2. Choose Chart→Format→Font or press Ctrl-F3. The Font dialogue appears. This is identical to the worksheet font dialogue; for more detailed information please see that section.
3. Make the desired modifications.

4. Click *OK* or press Return to exit, or click Menu to alter another format.

The colour *Automatic* as used here results in black text.

Text Background Colour

For the Legend, Value/Category Axis and Axis/Chart Title elements, you may choose the colour that all text for that element is written upon; the effect is that of a highlighter pen.

To alter the text background colour, perform the following steps:

1. Select the element containing the colour you wish to alter.
2. Choose Chart→Format→Text or press Ctrl-F5. The Text dialogue appears.
3. Select the new colour to use behind the text. The displayed colours are taken from Eureka's internal palette.
4. Click *OK* or press Return to exit, or click Menu to alter another format.

The colour *Automatic* as used here specifies no colour; the Fill colour used for that element shows through.

Note that to alter the colour of the text itself, you should choose Chart→Format→Font.

Number formatting

The Value and Category Axis elements can have their number formatting altered in just the same way as worksheet cells. It is by this method that, for example, a percentage chart can be created that displays 0% to 100% along the value axis instead of 0 to 1.

To alter the number formatting, perform the following steps:

1. Select the axis you wish to format.
2. Choose Chart→Format→Number or press Ctrl-F1. The Number dialogue appears.
3. Select the format from the scrollable list.
4. Click *OK* or press Return to exit, or click Menu to alter another format. You can also double-click on a number format to apply it.

The Number dialogue is identical to that used for worksheet number formatting. For details on the types of format and how to create custom formats, please refer to that section. Note that it refers to cells and worksheets throughout, though the principles are the same.

Altering Scales

Eureka chooses appropriate scales automatically. The range of the value (y) axis is set such that all the values to be displayed lie within that range.

If all the values are equal, this range will include zero; otherwise, zero is not necessarily displayed.

In addition, Eureka includes the facility to alter both Category and Value scales to your requirements.

To alter a scale, perform the following steps:

1. Select the Axis with the scale you wish to alter.
2. Choose Chart→Format→Scale or press Ctrl-F2. The Scale dialogue appears, tailored to the type of axis selected.
3. Make your modifications.
4. Click *OK* or press Return, or click Menu to alter another format.

The category axis may contain either numbers or labels. The Scale dialogue appears differently in these two cases.

For category axis labels, the dialogue lets you choose where the category axis crosses the value axis, how many categories to display between tick labels (in other words, to let you miss out every so many labels while still displaying the values), how many categories to display between tick marks, and whether to display categories in reverse order.

The first three of those options have a number entry field and an on/off button. By default, the fields display the current values calculated automatically. If you wish to change a value, you must remember to turn the option on for your entry to have an effect.

For numerical axes, the Scale dialogue lets you choose the range of the axis (ie. maximum and minimum values to display), the major and minor units (ie. the gap between each major or minor tick), where the axes cross, and whether to use a logarithmic scale or not.

Again, the entries only take effect if the option is switched on.

If a logarithmic scale is chosen, the major and minor unit options are greyed, and any negative or zero values are not plotted.

Moving the Legend

The Legend may be placed adjacent to any side of the chart itself. To move the legend, select Chart→Options→Legend (or press Ctrl-L) and select one of *Right*, *Bottom*, *Left* or *Top*, then click *OK* or press Return. The Legend may alternatively be hidden by selecting *None*.

Deleting Chart Elements

Some chart elements may be deleted, or removed from display.

Chart and Axis Title elements, the Legend and individual Series elements may be deleted by selecting the element and choosing Chart→Edit→Delete or pressing Del. The Legend may also be removed by choosing Chart→Options→Legend and selecting *None*. Series elements may also be deleted by choosing Chart→Edit→Series, selecting the series and clicking on *Delete*.

Axis labels may be hidden by selecting the axis, choosing Chart→Format→Pattern and selecting *None* from the Labels section.

Axes themselves may be hidden by choosing Chart→Options→Axes (or pressing Ctrl-A) and turning the appropriate button(s) off, then clicking *OK* or pressing Return.

Gridlines

Eureka allows you to display gridlines on charts in a variety of ways.

To display or remove gridlines, choose Chart→Options→Gridlines or press Ctrl-G. The Gridlines dialogue appears, allowing you to opt for major and/ or minor gridlines for either or both axes.

The major and minor gridlines appear as extensions to the major and minor ticks on the axes. This means that you can alter the colour and style of the gridlines using the Chart→Format→Pattern menu item after selecting the axis concerned.

Best-Fit Lines

It can be useful to calculate a best-fit line — a straight line that passes as close as possible to each of the data points, ideally passing through them all. Eureka can plot this line for you, in a number of ways.

To show the best-fit line for a data series, perform the following steps:

1. Chart your data. The data may have more than one series; one best-fit line is calculated per series.
2. Choose Chart→Options→Series or press Ctrl-E. The Series dialogue appears.
3. Pick the style of best-fit line you would like. These are:
   - Data, low to high X value — the best-fit line for each series is plotted between the lowest and highest X coordinate from all series in your data. If you change the x-axis scale, the best-fit lines do not extend past the edges of the data.
   - Chart, low to high X value — the best-fit line for each series extends to the edges of the chart, regardless of the positions of the data points.

- Intersect Y Axis — the best-fit line for each series extends to the y-axis.
- Intersect X Axis — the best-fit line for each series is extended until it intersects the x-axis.

You can also choose not to display the best-fit line.

There are also options whether or not to show the data points connected, and whether or not to plot the markers for each data point; these default to off and on respectively. Note that with both off, only the best-fit line (if shown) is displayed.

4. Click on *OK* or press Return to alter the chart display.

To produce best-fit lines for a single series in a multiple-series chart, choose the *Options* button in the Edit Series dialogue with that series selected.

Bar Spacing and Overlap

Eureka allows you to alter the relative positioning of bars within your chart, adjusting their sizes to fit.

To alter the positioning, choose Chart→Options→Bar Spacing. The Bar Spacing dialogue appears.

There are two settings: %Gap and %Overlap.

The %Gap setting determines the proportion of bar width left as space between each category along the axis. It defaults to 50.

The %Overlap setting determines the proportion of bar width between each series within a category. If this is negative, then the bars will overlap. The default is -25.

Thus, by default charts with more than one series contain 25% overlapping bars within a category, and half a bar's space between categories. If a chart only contains one series, then the %Overlap setting serves no purpose.

Click *OK* or press Return to confirm the changes.

Printing

Eureka allows you to print both worksheets and charts, and contains many options to help you get the best from your printer.

This section will describe Eureka's capabilities in full; in practice, a lot of the detail can be left out for Eureka's defaults to cover.

Set Print Area

By default Eureka prints your entire worksheet, up to its edges. However, if you are interested in only a small section of worksheet, there is a facility to mark that area alone for printing.

To do this, select the range you wish to print and choose Worksheet→Options→Set Print Area.

This defines a name, Print_Area, to refer to the selection you made. When you print, Eureka looks at this name to decide how much of the worksheet to print. In fact, Print_Area can be an arbitrary formula. If defined, it will designate the only printable area.

If you wish to revert to printing the whole worksheet, select the entire worksheet using the Select All button at the junction of the row and column headings and choose Worksheet→Options→Remove Print Area (which replaces the Set Print Area option). Alternatively you can delete the name Print_Area using the Define Name dialogue.

Set Print Titles

When printing a large worksheet, the output may extend onto more than one printed page. It can then become awkward determining the column or row heading your data belongs to, as this might appear only at the beginning of the worksheet.

Eureka helps you recognise your data by allowing you to define whole rows and/or columns as "print titles" which will appear on each page of your printout.

To do this, select the *whole* rows and/or columns to be used and choose Worksheet→Options→Set Print Titles.

This defines a name, Print_Titles, to refer to the selection you made. Typically it will look like "$1:$3,$C:$C" (so that rows 1:3 and column C appear on each printed page).

If you wish to cancel your print titles, select the entire worksheet using the Select All button at the junction of the row and column headings and choose Worksheet→Options→Remove Print Titles (which replaces the Set Print Titles option). You can alter the print titles by changing the definition of Print_Titles using the Define Name dialogue.

Set Page Break

By default Eureka chooses page breaks so that as much data as possible is printed on the page. You can, however, insert your own page breaks for more control over your output.

To insert a page break, perform the following steps:

1. Select the entire row or column you wish to be the first printed on a new page. Alternatively position the active cell just below and to the right of your desired horizontal and vertical page breaks.
2. Choose Worksheet→Options→Set Page Break. A horizontal and/or vertical line appears above and/or to the left of the selection, marking the place(s) to break. Page break lines are not printed on the paper.

Note that the area between two inserted page breaks will not necessarily appear on the same printed page; Eureka still breaks pages if there is too much data to fit. To force same-page printing use Page Setup's scaling functions.

Page breaks do not move their relative positions if columns or rows are inserted or deleted.

Remove Page Break

If you wish to remove a page break you have inserted, you can do so by positioning the active cell directly under or to the right of a page break line and choosing Worksheet→Options→Remove Page Break (this menu item replaces Set Page Break if a page break is already there).

To remove all page breaks, select the entire worksheet using the Select All button at the junction of the row and column headings and choose Worksheet→Options→Remove all Page Breaks.

Page Setup

To open the Page Setup dialogue, choose File→Page Setup from the Worksheet or Chart menus, or press shift-Print. From here you can customise your output. The dialogue is identical for both worksheets and charts, though the defaults in each case are different.

Page Setup information is saved along with the rest of the file.

If you click on *OK* or press Return, any changes you have made are kept and the dialogue is closed. If you click on *Print*, then the changes are kept and you proceed to the Print dialogue. Clicking *Cancel* will forget any changes.

Orientation

Here you specify whether your output should appear in Portrait or Landscape form. Choose Landscape if your data is wider than it is tall, or Portrait if it is taller than it is wide.

Both worksheet and chart printing default to Portrait setting.

Margins and Positioning

In this section of the Page Setup dialogue you can specify how much space to leave along the four sides of the paper. The units used in these entries depend on the setting of the environment variable Eureka$Units; this can be given the values mm, millimetres, cm or centimetres, with anything else resulting in inches (which is the default if Eureka$Units is not defined).

Here you can also choose to centre the printout horizontally and/or vertically along the page.

The default margins for charts and worksheets are 0.75 inches left and right, and 1 inch top and bottom.

By default, worksheets are not centred at all; charts are centred along both directions.

Headers and Footers

Eureka allows quite complex headers and footers to be generated, through use of special codes in the text. These are the codes and their meanings:

&l	Left justify the following text
&c	Centre the following text
&r	Right justify the following text
&f	Replace with the name of the file being printed
&d	Replace with the date in d/m/yy format
&D	Replace with the date in d-mmm-yy format
&t	Replace with the time
&p	Replace with the current page number
&n	Replace with the total number of pages
&y	Set the vertical offset from top or bottom of page in inches

The justification codes &l, &c and &r can be combined. For example, if the header text was the following:

&lLeft&rRight&cCentre&l again

the header would appear on the page as:

Left again   Centre   Right

with "Left again" and "Right" justified at either side of the page. Note that the "again" combined with the "Left" properly, even though they were apart in the text.

If the justified text is excessively long, it may overwrite other text in the printed line. This will not cause an error, but is not advisable for purely aesthetic reasons.

If no justification code is given, text is centred between the left and right margins.

The &y code is used to set the header and footer margins, which otherwise default to 0.5 inches. Use of this option helps with printers such as the HP Deskjet, whose absolute bottom margin appears to be 0.5 inches. When printing a footer with the default 0.5 inch margin at 150 DPI, the bottom-most pixel of a "y" descender is printed on a second page, rather than being clipped. If you have this problem, using &y0.52 in the footer resolves it.

Note that the &y setting must be given in inches and not the current Eureka$Units, as the code is saved as text and there is no indication of which units it was saved under.

For worksheets, the header defaults to "&f" and the footer to "Page &p". For charts, the header is again "&f" but the footer is left blank.

Page Size

This setting is currently ignored, but may be interpreted as a suggested page size when Eureka files are distributed.

The value defaults to A4 for both worksheets and charts.

Scaling output

In the Scale section of the Page Setup dialogue, you can specify custom enlargements or reductions of your output.

The *100%* option, which is the default for both worksheets and charts, prints normal size.

Selecting the *Custom* option allows you to enter your own scale factor in the adjacent edit field.

Alternatively, by selecting one or both of the *Fit To* options and editing their adjacent fields you can tell Eureka to scale the output to fit a number of pages across and/or down.

Other Page Setup Settings

The *Row & Column headings* button allows you to choose whether to include row numbers (1, 2, 3...) and column names (A, B, C...) in the output. This defaults to on for worksheets, and is greyed out for charts.

The *Black & White cells* button determines whether Eureka prints shades of grey to approximate to colours in multi-coloured worksheets, or to ignore colour information when printing. If it is on, then all text is printed black on white. This defaults to off for worksheets, and is greyed out for charts.

The *Gridlines* button, if set, tells Eureka to print the underlying grid of the worksheet. If this option is turned off, only defined borders are printed. It defaults to on for worksheets, and is greyed out for charts.

The Print Dialogue

To open the Print dialogue, either choose File→Print from the Worksheet or Chart menus, or press Print. Alternatively you can reach this dialogue via the Page Setup dialogue's *Print* button.

This dialogue has a small status bar, which displays your current printer settings.

Here you can choose to print all pages in your worksheet or a range of them, and you can specify the number of copies.

In addition you can call up the Page Setup dialogue by clicking on *Setup*.

Print dialogue information is saved along with the rest of the file.

When you are ready to print, click on *OK* or press Return.

Another dialogue will appear, signalling that printing is in progress. A status bar indicates how far the printing has proceeded. At any time you can press Escape to cancel the printing.

Printing by Dragging

You can print a saved worksheet or chart by dragging it to the printer driver icon. A dialogue will appear indicating how far the printing has proceeded. At any time you can press Escape to cancel the printing.

Text Extending into Neighbouring Cells

If a cell contains a long text string, then the string may overrun into neighbouring cells. This may look fine on screen, but can cause problems when printing.

Consider this minimalist example of a worksheet:

	A	B
1	This year's Profits	
2	£2.50	

The data is contained only in cells A1 and A2; but the text extends to cell B1. By default, Eureka will print up to the edges of the used part of the worksheet — ie those cells containing data.

Thus the printed worksheet will resemble:

	A
1	This year'
2	£2.50

To get around this problem, use Worksheet→Options→Set Print Area to include those cells which are inadvertently covered — here you would select the range A1:B2. Alternatively you can increase the width of column A to fit the text, by double clicking with the mouse at the junction of the A and B column headings at the top of the worksheet.

Formula Find and Replace

Eureka allows you to search through your worksheet for arbitrary text, optionally replacing it. In addition you may find and replace formulas.

To find or replace text or formulas within your worksheet, perform the following steps:

1. If you want to restrict the search to a particular area, position the current cell at the start position.
2. Choose Worksheet→Formula→Find or Worksheet→Formula→Replace. The Find or Replace dialogue will appear.
3. Enter the text or formula to be found (and that to replace it if applicable) and set the options to your requirements. The search text may contain wildcards * and ?. The wildcards may be suppressed with ~. You can choose to limit the search to values only, formulas only, or both. Also you can restrict the search to the active cell and below, or search the entire sheet. Searching can be ordered rows first or columns

first. There is a Case Sensitive option to allow more precise matching.

4. Click *OK* or press Return.
5. Another dialogue will appear. The current cell will be moved to the first occurrence, if any, of the search string. You can click on *Find Next* to continue the search, or on *Close* to stop searching. If you are replacing, there are also options to replace this occurrence and continue, to replace all occurrences, or to replace this occurrence and then stop.
6. When Eureka can find no more occurrences, a message indicating the number of occurrences found will appear. Click on *Close* to leave the dialogue.

Note that the values that Eureka searches through are the underlying representations of the data displayed in the cells — the values shown in the formula bar. That is, all formats are ignored. For example, if you want to change all occurrences of the pound sign to the dollar sign, searching will only find those directly entered as text. In this case the answer is to use a custom number format. Times and dates are treated slightly differently; please see that section for further information.

Calculation is temporarily disabled during replace. If you replace within a formula, the cell will be displayed as zero. Likewise, replacing within a precedent cell does not cause calculation of its dependents. When you dismiss the Replace action dialogue, Eureka calculates the sheet if it is set for automatic recalculation.

The Simplesum Icon

Eureka's Simplesum icon makes a 'best guess' for the data you wish to sum. It looks above and to the left of the active cell, stopping at formulas or labels. Usually it picks the correct selection for the marquee.

If you want to change its selection, you can do so by typing (the SUM function is inserted into the formula bar with its current range), by clicking Adjust to change the range (the anchor for the marquee remains adjacent to the active cell), or by clicking and dragging a new range with Select.

Note that Simplesum stops at formulas; you can still use it however to insert the SUM function and allow you to point to the range you wish.

Range Selection

You can drag out multiple selections by holding down the Ctrl key as you make them. Once they are made, you can modify them without losing the entire multiple selection by using the function keys.

- Ctrl-Shift-F10 collapses the *current* selection in a multiple selection to a single cell. This is equivalent to clicking Adjust on the active cell.
- F11 moves the active cell to the corners of a selection, in a clockwise fashion. Shift-F11 does the same anti-clockwise. This can be especially useful when adjusting the extent of a selection from the keyboard, after the selection has been completed.
- Ctrl-F11 moves the active cell to the first cell of the *next* selection in a multiple selection. Ctrl-Shift-F11 moves the active cell to the first cell of the *previous* selection in a multiple selection. Selection components are numbered as they are made by the user, and which selection is 'next' or 'previous' is based on this numbering.
- Within a selection the following keys lock in the entry and move the cell pointer, *without* cancelling the selection.
  - Return, Ctrl-Tab: move the active cell down a row, wrapping around to the top of the selection when at the bottom.
  - Tab: moves the active cell to the right, wrapping around to the left side of the selection when at the right.
  - Shift-Tab: moves the active cell to the left, wrapping around to the right side of the selection when at the left.
  - Ctrl-Shift-Tab: moves the active cell up a row, wrapping around to the bottom of the selection when at the top.

  When at the top-left or bottom-right of a selection, these keys move to the bottom-right of the previous or top-left of the next selection in a multiple selection as you would expect.

Naming using 'Create Names'

In addition to defining names using Worksheet→Formula→Define Name, Eureka lets you define names in bulk if they can be selected as a range.

For example, with this worksheet:

	A	B	C	D	E	F
1		Mon	Tue	Wed	Thu	Fri
2	Foo	5	2	3	1	4
3	Bar	7	3	6	3	2
4	Baz	8	4	6	1	4

You might like to define names Foo, Bar and Baz to refer to the figures in the row. Also, the names Mon, Tue, Wed, Thu and Fri might usefully be defined for the column figures.

This can be done quickly and easily with the following method:

1. Select the range you wish to define names for.
2. Choose Worksheet→Formula→Create Names. The Create Names dialogue appears.
3. Set the options according to the location of the names to define — they can be in the top row, left column, bottom row and/or right column.
4. Click *OK* or press Return.

In the example above, you would select the range A1:F4 and choose *Top Row* and *Left Column* in the Create Names dialogue. Eight names would be defined at once. Note that Eureka does not define a name for the empty cell A1.

If a cell contains spaces, then those spaces are replaced with underscores in the name.

If a name conflicts with an existing name, a dialogue appears allowing you to choose between the existing definition and the new one.

Paste Name List

It can be useful for documentation purposes to have a list of every name in the worksheet and its definition. Eureka allows you to do this using a single menu item.

Simply position the current cell in a blank area of your worksheet and choose Worksheet→Formula→Paste Name List.

The names and their definitions will be entered into two columns down the worksheet from the current cell.

If a blank cell lies to the right of a name, then that name has been used in the worksheet (for example in a formula "=Profits") but has not yet been defined. These "tentative" names do not appear in the Define Name or Goto dialogues.

A dialogue will appear if the name list will overwrite existing cells, allowing you to go ahead or cancel the paste.

Goto

Eureka allows you to go quickly to a particular cell without having to scroll, using the Goto dialogue. It also lets you go to a named area.

To jump to a cell, perform the following steps:

1. Choose Worksheet→Formula→Goto or press F5. The Goto dialogue appears.

2. Either type the cell reference in the edit field, or select a name from the scrollable list.
3. Click *OK* or press Return. Alternatively you may double-click on a name to choose it and leave the dialogue.

Eureka will jump to the chosen cell or range of cells, which will become the current selection.

Goto works with formulas returning multiple selections. For example, make a multiple selection and then choose Worksheet→Options→Set Print Area. Open the Goto dialogue, and since the Print_Area name is not a simple selection you will need to click on *Show All Names* to display it. Then double-click on Print_Area; the original multiple selection will become selected, and the active cell will be positioned in the first cell of the first selection you made.

You can use Goto in two ways to alter the current selection.

- The first method extends the selection to include the bottom-right corner of the range chosen using the Goto dialogue. To do this, make your initial selection then choose Goto and pick another cell or name. Leave Goto by shift-clicking *OK* or pressing shift-Return.
- The second method makes a multiple selection using the current selection (which may itself be multiple) and the range chosen using the Goto dialogue. To do this, make your initial selection(s) then choose Goto and pick another cell or name. Leave Goto by Ctrl-clicking *OK* or pressing Ctrl-Return.

Altering the Worksheet Display

Eureka allows you to view the worksheet in other forms using the Worksheet→Options→Window menu item.

The Window dialogue which appears when you choose this item lets you opt to remove or include gridlines, row and column headings, zero values (allowing you to hide cells which contain zero) or display formulas instead of their results.

Click *OK* or press Return to exit the dialogue.

Displaying Formulas in the Worksheet

Normally Eureka displays only the results of formulas, not the formulas themselves. However, it may be useful for documentation purposes to view the mechanisms by which a worksheet derives its results.

This can be done quickly and easily with the following method:

1. Select the range you wish to define names for.
2. Choose Worksheet→Formula→Create Names. The Create Names dialogue appears.
3. Set the options according to the location of the names to define — they can be in the top row, left column, bottom row and/or right column.
4. Click *OK* or press Return.

In the example above, you would select the range A1:F4 and choose *Top Row* and *Left Column* in the Create Names dialogue. Eight names would be defined at once. Note that Eureka does not define a name for the empty cell A1.

If a cell contains spaces, then those spaces are replaced with underscores in the name.

If a name conflicts with an existing name, a dialogue appears allowing you to choose between the existing definition and the new one.

Paste Name List

It can be useful for documentation purposes to have a list of every name in the worksheet and its definition. Eureka allows you to do this using a single menu item.

Simply position the current cell in a blank area of your worksheet and choose Worksheet→Formula→Paste Name List.

The names and their definitions will be entered into two columns down the worksheet from the current cell.

If a blank cell lies to the right of a name, then that name has been used in the worksheet (for example in a formula "=Profits") but has not yet been defined. These "tentative" names do not appear in the Define Name or Goto dialogues.

A dialogue will appear if the name list will overwrite existing cells, allowing you to go ahead or cancel the paste.

Goto

Eureka allows you to go quickly to a particular cell without having to scroll, using the Goto dialogue. It also lets you go to a named area.

To jump to a cell, perform the following steps:

1. Choose Worksheet→Formula→Goto or press F5. The Goto dialogue appears.

2. Either type the cell reference in the edit field, or select a name from the scrollable list.
3. Click *OK* or press Return. Alternatively you may double-click on a name to choose it and leave the dialogue.

Eureka will jump to the chosen cell or range of cells, which will become the current selection.

Goto works with formulas returning multiple selections. For example, make a multiple selection and then choose Worksheet→Options→Set Print Area. Open the Goto dialogue, and since the Print_Area name is not a simple selection you will need to click on *Show All Names* to display it. Then double-click on Print_Area; the original multiple selection will become selected, and the active cell will be positioned in the first cell of the first selection you made.

You can use Goto in two ways to alter the current selection.

- The first method extends the selection to include the bottom-right corner of the range chosen using the Goto dialogue. To do this, make your initial selection then choose Goto and pick another cell or name. Leave Goto by shift-clicking *OK* or pressing shift-Return.
- The second method makes a multiple selection using the current selection (which may itself be multiple) and the range chosen using the Goto dialogue. To do this, make your initial selection(s) then choose Goto and pick another cell or name. Leave Goto by Ctrl-clicking *OK* or pressing Ctrl-Return.

Altering the Worksheet Display

Eureka allows you to view the worksheet in other forms using the Worksheet→Options→Window menu item.

The Window dialogue which appears when you choose this item lets you opt to remove or include gridlines, row and column headings, zero values (allowing you to hide cells which contain zero) or display formulas instead of their results.

Click *OK* or press Return to exit the dialogue.

Displaying Formulas in the Worksheet

Normally Eureka displays only the results of formulas, not the formulas themselves. However, it may be useful for documentation purposes to view the mechanisms by which a worksheet derives its results.

To display the formulas, choose Worksheet→Options→Window and select the *Formulas* option. Then Click *OK* or press Return.

Eureka alters the display to allow for the fact that formulas are usually longer than their results. Column widths are doubled, and cell contents are prevented from spilling outside the cell borders.

Worksheet Zoom

Eureka allows you to magnify the worksheet display, or reduce it slightly.

To do this, perform the following steps:

1. Choose Worksheet→Window→Zoom. The Zoom dialogue appears.
2. Select from the predefined zoom options or choose your own using *Custom* and the arrows (clicking an arrow changes the figure by 1%, shift-clicking changes by 10%). You can also pick *Fit Selection* to make the current selection fill the display.
3. Click *OK* or press Return. Alternatively you can double-click on an option to choose it and leave the dialogue.

Valid zoom scale factors lie between 20% and 800%.

The *Fit Selection* button takes the union of the current selection, and scales it to fit the entire window, ignoring panes. If you have selected whole rows or columns, then Eureka computes the intersection of the range and the active area, and uses this in computing the union of the selections.

As a special case, if only a single range is selected, and it is a whole row or whole column selection, then a row selection is scaled so the rows fill the window, and vice versa for column selections.

Saving sections of your Worksheet

Eureka allows you to limit saves to the current selection.

Multiple selections are possible for normal and Lotus file types. All cells within the selection are saved, as is the formatting for rows and columns intersecting the selection. The cells in the saved sheet appear in the same positions as in the original.

CSV and SID files also permit saving multiple selections; however, care is needed as the individual ranges are output in the order they were selected.

Drawfile saves are limited to the first range in a multiple selection.

To save a part of your worksheet, make the selection and choose Worksheet→File→Save As or press F3. The Save As dialogue appears. Click on *Limit To Selection*, change the filename to suit and click *OK*, press Return or drag the icon to the correct directory viewer.

CSV and SID Files

Eureka can save and load both CSV (Comma Separated Values) and SID (Software Independent Data format) files. These are standard formats which allow data to be imported and exported between applications.

A CSV file contains lines (rows of the spreadsheet) with data items separated by commas (indentifying the columns). Text is contained in double quotes within a CSV file, so that commas in the text are not interpreted as data separators.

SID files are similar to CSV files, using commas to separate data. However, lines are terminated with CR-LF (CSV files use just LF) and, when saving, Eureka does not place quotes around text in any line whose first entry begins with %%. These lines are regarded as SID commands.

Note that only the underlying data is saved in CSV and SID files; for instance, times and dates are stored as serial numbers (see Times and Dates, below) and will be saved as such.

Templates

Templates are very similar to ordinary files. The difference is that when creating a new document via the Templates dialogue, files are named "Sheet2", "Sheet3" etc. just like new, blank documents.

They are used for creating new blank documents with custom predefined styles, formatting information, number formats and so on.

To save a worksheet as a template, choose Worksheet→File→Save As (or press F3) and select the Template file type from the scrollable list. Edit the name to your requirements and click *OK* or press Return, or alternatively drag the icon to the <Eureka$Dir>.Templates directory.

To create a new document using an existing template, choose Eureka→New Document→Templates (or just click Adjust on the Eureka icon on the Icon Bar). A scrollable list of templates appears, sorted by file type and then alphabetically. There can be as many as 64 templates. Select the required template and click *OK* or press Return, or double click on the template name.

If you want to edit a template file itself, then open the Template dialogue as before but hold down shift when clicking *OK*, pressing Return or double-clicking on a template file. Alternatively you can drag a template icon from the <Eureka$Dir>.Templates directory.

The Icon Bar

The Eureka icon on the Icon Bar can be used in three ways.

- If you click Menu on it, the Eureka menu appears as if you had clicked Menu on the control panel.
- Clicking Adjust on the icon allows you to open a template as a new document, as if you had selected Eureka→New Document→Templates.
- Shift-clicking on the icon restores the control panel if it is hidden. If it is showing, then the active child window is moved to the front followed by the control panel itself.

Times and Dates

Eureka handles times and dates by coding them as a number. This serial number has an integer part and a fractional part; the integer holds the number of days since the beginning of the twentieth century, and the fractional part is the time of day.

Valid dates range from 1 January 1600 to 31 December 3199. Dates before 1900 will have negative serial numbers.

To display a serial number as a time and date, you change the number format for the cell it is in using Worksheet→Format→Number or Ctrl-F1. Helpfully, if you want to enter a date or time directly into the worksheet, Eureka allows you. It recognises all the built-in number formats for times and dates and converts automatically, changing the number format of the cell.

Once recognised, Eureka displays dates and/or times in the formula bar in a standard form. Dates conform to the pattern d/m/yyyy, and times to h:mm:ss or h:mm depending on the original entry. Times always follow dates if both are present. This means that the Find and Replace functions can find, for example, all references to "/1993" without trouble.

Automatic Format Recognition

When you click Accept or press Return after entering data in the formula bar, Eureka examines the data to see whether it conforms to any of the built-in number formats. If it does, Eureka converts the data to numerical form, and alters the number format for that cell so that the data is displayed as it was typed.

This means that the round-about route (enter data numerically, change the number format by hand) can be circumvented just by typing the data in the format you require.

Special Functions for the Formula Bar

When typing in the formula bar, there are some special functions you can invoke. If the text caret is within a cell or cell range reference, for example A1 or B2:C4, then the F4 key cycles through the absolute/relative addressing forms for that reference.

Most menu items are disabled when typing in the formula bar. Two that are still available are Worksheet→Formula→Paste Function and Worksheet→Formula→Paste Name, which have accelerator keys Ctrl-F and Ctrl-N respectively. These keys only have this effect while in the formula bar.

Pasting Functions and Names

Eureka contains over 150 functions, and you can define many names yourself. These can be pasted into the formula bar using these two menu items.

To paste a function into the formula bar, perform the following steps:

1. Type the formula up to the point where you require the function; if it is to appear at the start, you need not type anything.
2. Choose Worksheet→Formula→Paste Function. The Paste Function dialogue appears.
3. Select the category your function appears under, from the list on the left. If you are not sure, then select All.
4. Select the function from the list which appears on the right.
5. You can now select either *Paste* or *Paste Arguments*. The latter includes helpful indicators of the purpose of each of the function's arguments, which you will need to replace, whereas the former leaves the arguments blank.
6. The function will be pasted, and the caret will return to the formula bar so you can continue typing.

To paste a name into the formula bar, perform the following steps:

1. Type the formula up to the point where you require the name; if it is to appear at the start, you need not type anything.

2. Choose Worksheet→Formula→Paste Name. The Paste Name dialogue appears.
3. Select the name you require from the list.
4. Click *OK* or press Return. Alternatively you can double-click on the name you require.
5. The name will be pasted, and the caret will return to the formula bar so you can continue typing.

Program Startup

Eureka can be configured to either open a blank worksheet on startup, or to just place its icon upon the Icon Bar. To produce the former behaviour, set the environment variable Eureka$OpenBlankWorksheetOnBoot to 1. If this is unset or set to any other value, the latter behaviour will result.

Note that opening of a blank sheet will only occur provided the command line was empty (ie. Eureka was not started by double-clicking on a Eureka document).

The KeyStrip

Eureka is distributed with a drawfile named KeyStrip. This contains a handy guide to the short cuts obtainable from function keys.

It can be printed simply by dragging the file to the printer icon.

Altering the Palette

In addition to choosing the colours for elements of your worksheets and charts from a predefined set, you can alter the palette for complete customisation.

To alter the palette used for a chart or worksheet, perform the following steps:

1. Choose Options→Palette from the Worksheet or Chart menu as applicable. The Palette dialogue appears.
2. Click on the colour you wish to edit, then click on *Edit*. The Colour Picker dialogue appears.

   The lower area initially shows the details of the colour selected: its Hue, Saturation and Luminance ratios and Red, Green and Blue ratios (the latter in both value and slider forms), as well as the colour itself in the New/Original box. As you alter the ratios by editing the values, clicking the arrows or moving the sliders, the top half of the New/ Original box changes to reflect the new colour; the original is still displayed beneath. You can click in this box to restore the original colour.

The top of the dialogue allows you to select colours without bothering with ratios. There is a colour wheel, which contains white at its centre mixed in decreasing proportions to the primary colours and their blends at its circumference. As you select a colour by clicking in the wheel, then the mixing bar alongside updates to show all the shades between that colour and black. Two arrow markers either side of the mixing bar indicate the new colour displayed in the New/Original box. The markers can be moved by clicking within the bar.

In both the wheel and the bar, the number of shades displayed depends on the number of colours available in the screen mode used for Eureka.

3. When you are satisfied with the new colour, click on *OK* or press Return. To exit the dialogue without altering the colour, click on *Cancel*.
4. You now return to the Palette dialogue, where you can edit another colour by the same process.
5. To accept all the colour changes, click on *OK* or press Return. To ignore them all, click on *Cancel*.

Note that the colour changes will only take effect when you accept them from the Palette dialogue.

Each worksheet and chart has its own palette.

Remember that by default some colours (eg. for fonts) are specified as *Automatic*, and some (for instance a Chart Background element's Fill pattern) are set to *None*. If an expected colour change does not happen, then this may be the reason.

Copying the Palette

Because each worksheet and chart can have its own palette, Eureka allows you to copy complete palettes from one to another without having to make manual changes.

To do this, perform the following steps:

1. Ensure that the worksheets or charts involved in the palette copying are both open.
2. Choose Options→Palette (from the Worksheet or Chart menu as applicable) for the palette you wish to change. The Palette dialogue appears.

3. From the drop-down list select the name of the worksheet or chart you wish to copy the palette from; the colour boxes will be updated to reflect the new palette.
4. Click on *OK* or press Return to accept the new palette, or click on *Cancel* to ignore the changes.

Note that the worksheet or chart will reflect any palette changes only upon leaving the Palette dialogue.

#AND# and #OR#

In addition to the functions AND() and OR() given in the function reference section, Eureka contains two extra logical operators #AND# and #OR# which serve the same purpose but reduce the amount of nesting needed, simultaneously increasing readability.

For instance, the formula '=AND(A1=1,OR(B1<1,C1>1))' can be rewritten '=A1=1#AND#(B1<1#OR#C1>1)'. Note that #AND# and #OR# have equal precedence, below that of the comparison operators.

ABS

Syntax

ABS(*number*)

Description

ABS returns the absolute value of *number*.

Example

`ABS(2)` equals 2.

`ABS(-2)` equals 2.

See also

SIGN

ACOS

Syntax

ACOS(*number*)

Description

ACOS returns the arccosine of *number*, the angle whose cosine is *number*. The returned angle is in the range (0,*pi*).

Number should be a value in the range (-1,1).

Example

`ACOS(0)` equals *pi*/2.

`DEGREES(ACOS(-0.5))` equals 120.

See also

COS

ACOSH

Syntax

ACOSH(*number*)

Description

ACOSH returns the inverse hyperbolic cosine of *number*.

Number should be ≥ 1.

Example

`ACOSH(1)` equals 0.

`ACOSH(2)` equals 1.317.

See also

COSH

AND

Syntax

AND(*logical1*,*logical2*,...*logical14***)**

Description

AND returns TRUE if all its arguments evaluate TRUE. If one or more evaluate FALSE, AND returns FALSE.

AND accepts 1-14 arguments, which should be logical values, or references or arrays containing logical values. AND ignores blank cells and numbers and text within references and arrays. However, for the purposes of AND, non-zero numbers given as arguments are considered equivalent to TRUE, and zero given as an argument is considered equivalent to FALSE. A reference argument may be a range as well as a single cell. If no logicals are supplied, AND returns the #VALUE! error value.

Example

`AND(A1>=1,A1<=6)` equals TRUE if A1 contains a number in the range (1,6).

`IF(AND(A1>=1,A1<=6),"","Number is out of range")` equals the empty text, "", if A1 is in the range (1,6), and "Number is out of range" if A1 contains a number outside this range.

`AND(A1:A3)` equals FALSE if A1:A3 contains the values TRUE, FALSE, and TRUE.

See also

OR, NOT, IF

ASIN

Syntax

ASIN(*number*)

Description

ASIN returns the arcsine of *number*, the angle whose sine is *number*. The returned angle is in the range (*-pi*/2,*pi*/2).

Number should be a value in the range (-1,1).

Example

`ASIN(-1)` equals *-pi*/2.

`DEGREES(ASIN(0.5))` equals 30.

See also

SIN

ASINH

Syntax

ASINH(*number*)

Description

ASINH returns the inverse hyperbolic sine of *number*.

Example

`ASINH(0)` equals 0.

`ASINH(2)` equals 1.4436.

See also

SINH

ATAN

Syntax

ATAN(*number*)

Description

ATAN returns the arctangent of *number*, the angle whose tangent is *number*. The returned angle is in the range *(-pi/2,pi/2)*.

Number should be a value in the range (-1,1).

Example

`ATAN(1)` equals *pi*/4.

`DEGREES(ATAN(-1))` equals -45.

See also

TAN

ATAN2

Syntax

ATAN2(*x_number,y_number*)

Description

ATAN2 returns the arctangent of the coordinate *(x_number,y_number)*, which is the angle between the *x*-axis and a line extending from the origin to the specified point. The returned angle is in the range [*-pi,pi*]; however, if both arguments equal zero, ATAN2 returns the #INV! error value.

Example

`ATAN2(1,1)` equals *pi*/4.

`ATAN2(1,-1)` equals -*pi*/4.

`ATAN2(-1,1)` equals 3*pi*/4.

See also

ATAN, TAN

ATANH

Syntax

ATANH(*number*)

Description

ATANH returns the inverse hyperbolic tangent of *number*.

Number should satisfy $-1 < number < 1$.

Example

`ATANH(0)` equals 0.

`ATANH(0.5)` equals 0.5493.

See also

TANH

AVEDEV

Syntax

AVEDEV(*number1*,*number2*,...*number14*)

Description

AVEDEV computes the mean absolute deviation of its arguments.

AVEDEV accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. AVEDEV ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, AVEDEV returns the #DIV/0! error value.

AVEDEV is computed via the following formula:

$$\text{AVEDEV} = \frac{\sum |x - \bar{x}|}{n}$$

Example

`AVEDEV(5,6,7)` equals 0.6667.

`AVEDEV(A1:A3)` equals 0.6667 if A1:A3 contains the values 5, 6, and 7.

See also

STDEV, STDEVP, VAR, VARP

AVERAGE

Syntax

AVERAGE(*number1*,*number2*,...*number14*)

Description

AVERAGE computes the arithmetic mean of its arguments.

AVERAGE accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. AVERAGE ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, AVERAGE returns the #DIV/0! error value.

Example

`AVERAGE(5,6,7)` equals 6.

`AVERAGE(A1:A3)` equals 6 if A1:A3 contains the values 5, 6, and 7.

See also

SUM, PRODUCT

CEILING

Syntax

CEILING(*number*,*significance*)

Description

CEILING rounds *number* up to the nearest multiple of *significance*. Both arguments to CEILING must have the same sign; otherwise, CEILING returns the #INV! error value.

Example

`CEILING(1.25,0.2)` equals 1.4.

`CEILING(-2.4,-0.5)` equals -2.5.

See also

FLOOR

CHAR

Syntax

CHAR(*number*)

Description

CHAR returns the character corresponding to *number*.

Number should be an ASCII value in the range 1-255.

Example

`CHAR(65)` equals "A".

See also

CODE, CLEAN

CHOOSE

Syntax

CHOOSE(*index_num,value1*,*value1, value2,...value13***)**

Description

CHOOSE selects one of its *value* arguments based upon the value of *index_num* and returns this argument.

Index_num is the 1-based index into the remaining arguments. If *index_num* is < 1 or greater than the number of remaining arguments, CHOOSE returns the #INV! error value.

The *value* arguments can be any of Eureka's data types. If the selected argument is a reference, CHOOSE returns that reference, meaning CHOOSE can be used in places where a reference is required, such as on either side of the range operator, ':'. If the selected argument is an array, CHOOSE returns that array.

If *index_num* is an array, then CHOOSE returns the selected arguments in an array.

Example

`CHOOSE(2,1,"text",3)` equals "text".

`AVERAGE(CHOOSE(A1,B1:B10,C1:C10))` equals `AVERAGE(C1:C10)` if A1 contains 2.

See also

HLOOKUP, INDEX, LOOKUP, MATCH, VLOOKUP

CLEAN

Syntax

CLEAN(*text*)

Description

CLEAN removes non-printing characters (control codes) from *text*.

Example

`CLEAN("a"&CHAR(10)&"b")` equals "ab".

See also

PROPER, TRIM

CODE

Syntax

CODE(*text*)

Description

CODE returns the numeric (ASCII) value of the first character in *text*.

Example

`CODE("A")` equals 65.

`CODE("ABC")` equals 65.

See also

CHAR

COLUMN

Syntax

COLUMN(*reference***)**

Description

COLUMN returns the column number of *reference*, an integer in the range [1,256]. If *reference* is omitted, COLUMN returns the column number of the cell containing the formula. If *reference* is for a range, and the formula is entered as an array formula, COLUMN returns the column numbers of *reference* as a horizontal array.

COLUMN returns the #INV! error value if *reference* is a multiple area or not a reference.

Example

`COLUMN()` equals 1 if used in a formula in column A of a worksheet.

`COLUMN(C3)` equals 3.

`{COLUMN(A1:C5)}` equals {1,2,3}.

See also

COLUMNS, ROW, ROWS

COLUMNS

Syntax

COLUMNS(*array***)**

Description

COLUMNS returns the number of columns in an array or reference.

COLUMNS returns the #INV! error value if *array* is a multiple area or not an array or reference.

Example

`COLUMNS(A1:B5)` equals 2.

`COLUMNS({1,2,3;4,5,6})` equals 3.

See also

COLUMN, ROW, ROWS

COMBIN

Syntax

COMBIN(*number_of_items,number_chosen*)

Description

COMBIN returns the number of combinations of size *number_chosen* taken from a set containing *number_of_items* members. A combination is an unordered collection of items. If n = *number_of_items* and k = *number_chosen*, then the formula for COMBIN is:

$$C_{k,n} = \frac{n!}{k!(n-k)!}$$

Example

`COMBIN(5,3)` equals 10.

`COMBIN(5,5)` equals 1.

`COMBIN(10,4)` equals 210.

See also

PERMUT

CORREL

Syntax

CORREL(*x_array,y_array*)

Description

CORREL returns the correlation coefficient of the data sets *x_array* and *y_array*.

X_array and *y_array* should be arrays or references with identical dimensions. CORREL ignores blank cells and text and logical values within references and arrays. A number occurring in one array must be matched by a number at the same position in the complementary array; otherwise, CORREL returns the #VALUE! error value. CORREL returns #DIV/0! if the standard deviation of either *x_array* or *y_array* is zero.

The formula for CORREL is:

$$\rho_{xy} = \frac{\mathrm{Cov}(X, Y)}{\sigma_x \sigma_y}$$

Example

`CORREL({1,4,-3,8,10},{2,6,7,10,16})` equals 0.740413.

See also

COVAR

COS

Syntax

COS(*angle_in_radians*)

Description

COS returns the cosine of *angle_in_radians*.

Example

`COS(0)` equals 1.

`COS(PI()/2)` equals 0.

See also

ACOS, SIN, TAN

COSH

Syntax

COSH(*number*)

Description

COSH returns the hyperbolic cosine of *number*. COSH is defined by the formula:

$$\cosh\ x = \frac{e^{x} + e^{-x}}{2}$$

Example

`COSH(0)` equals 1.

`COSH(2)` equals 3.7622.

See also

ACOSH

COUNT

Syntax

COUNT(*number1*,*number2*,...*number14*)

Description

COUNT returns a count of how many numbers occur within its arguments.

COUNT accepts 1-14 arguments. An argument will be counted if it is a number, logical value, text representation of a number, or a null argument. If an argument is an array or reference, only numbers within that array or reference are counted; COUNT ignores all other data types within arrays and references. In particular, error values do not cause COUNT to fail.

Example

`COUNT(A1:A2,1,FALSE,"22-Sep-92",#DIV/0!)` equals 4 if A1 contains 2 and A2 contains “text”.

See also

COUNTA

COUNTA

Syntax

COUNTA(*number1*,*number2*,...*number14*)

Description

COUNTA returns a count of how many values occur within its arguments. Unlike COUNT, COUNTA includes non-blank values, such as text that cannot be converted to numbers and error values.

COUNTA accepts 1-14 arguments. All arguments supplied to COUNTA are counted. However, if an argument is an array or reference, empty cells within that array or reference are excluded from the count.

Example

`COUNTA(A1:A2,1,FALSE,"22-Sep-92","text",#DIV/0!)`
equals 6 if A1 contains "text" and A2 is empty.

See also

COUNT

COVAR

Syntax

COVAR(*x_array,y_array*)

Description

COVAR returns the covariance of the data sets *x_array* and *y_array*.

X_array and *y_array* should be arrays or references with identical dimensions. COVAR ignores blank cells and text and logical values within references and arrays. A number occurring in one array must be matched by a number at the same position in the complementary array; otherwise, COVAR returns the #VALUE! error value. COVAR returns #DIV/0! if the data sets are empty.

The formula for COVAR is:

$$\mathrm{Cov}(X, Y) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)$$

Example

`COVAR({1,4,-3,8,10},{2,6,7,10,16})` equals 16.2.

See also

CORREL

DATE

Syntax

DATE(*year,month,day*)

Description

DATE returns the serial number of the indicated date.

Year should be a number in the range 0-222, in which case DATE adds 1900 to it, or 1900-2222, in which case it is used directly.

Month should be a number in the range 1-12.

Day should be a number in the range 1-31.

DATE returns #INV! if any of its arguments are less than 1; however, if *day* or *month* exceed their respective ranges, DATE *will* function properly, in effect "rolling over" into successive months and years.

Example

`DATE(92,9,22)` equals 33869, the serial number corresponding to September 22, 1992.

`DATE(91,2,29)` equals 33298, the serial number corresponding to March 1, 1991, because 1991 is not a leap year.

See also

TIME

DATEVALUE

Syntax

DATEVALUE(*date_text*)

Description

DATEVALUE returns the serial number corresponding to *date_text*, which should be a textual date in one of Eureka's built-in date formats. These formats are also recognised when textual dates are used directly in formulas and when entered into cells.

Example

`DATEVALUE("22-Sep-92")` equals 33869.

`DATEVALUE(A1)` equals 33869 if A1 contains the text ="22-Sep-92".

`"22-Sep-92"+1` equals 33870.

`A1+1` equals #VALUE! if A1 contains the text "22-Sep-92", but `DATEVALUE(A1)+1` equals 33870.

See also

TIMEVALUE, VALUE

DAY

Syntax

DAY(*serial_number*)

Description

DAY returns the day of the month of *serial_number*, an integer falling in the range 1-31.

You may use a textual date rather than a number; Eureka converts the text to a serial number automatically. However, text in references and arrays is *not* converted and generates a #VALUE! error when used with DAY.

Example

`DAY("22-Sep-92")` equals 22.

`DAY(33000.34)` equals 7.

See also

MONTH, YEAR, HOUR, MINUTE, SECOND

DAYS360

Syntax

DAYS360(*start_date,end_date*)

Description

DAYS360 returns the number of days between *end_date* and *start_date* based upon a 360 day (twelve 30 day months) year.

Example

`DAYS360("30-Oct-92","1-Nov-92")` equals 1.

`"1-Nov-92"-"30-Oct-92"` equals 2.

See also

DAY

DAYSPERMONTH

Syntax

DAYSPERMONTH(*serial_number*)

Description

DAYSPERMONTH returns the number of days in the month of *serial_number*.

You may use a textual date rather than a number; Eureka converts the text to a serial number automatically. However, text in references and arrays is *not* converted and generates a #VALUE! error when used with DAYSPERMONTH.

Example

`DAYSPERMONTH("22-Sep-92")` equals 30.

`DAYSPERMONTH("1-Feb-92")` equals 29.

`DAYSPERMONTH("1-Feb-2001")` equals 28.

See also

DAY

DDB

Syntax

DDB(*cost,salvage,life,period,* *factor*)

Description

DDB returns the depreciation of an asset for the indicated period using the double-declining balance method.

Cost is the inital cost of the asset.

Salvage is the salvage value, the amount the asset will be worth after all depreciation has been taken. DDB adjusts the depreciation for the final period as necessary so as to avoid depreciating an asset below its salvage value.

Life is the number of periods over which the asset will be depreciated.

Period is the period for which you wish the depreciation.

Factor controls the rate at which the balance declines. If you omit *factor*, DDB considers it equal to 2, resulting in a double-declining balance calculation.

The formula for DDB is:

DDB = ((*cost-accumulated depreciation*)**factor*)/*life*.

Example

Suppose you have purchased a piece of equipment for £15,000, and it is to be depreciated over 10 years. If the salvage value is £4,000, then the first year's depreciation is:

`DDB(15000,4000,10,1)` which equals 3000.

The sixth year's depreciation is:

`DDB(15000,4000,10,6)` which equals 915.2.

If you continue with this example, you will find that DDB forces depreciation for all years after the sixth to zero, because the asset was fully depreciated by the sixth year, and DDB will not depreciate an asset below its salvage value.

See also

DB, SLN, SYD, VDB

DEGREES

Syntax

DEGREES(*angle_in_radians*)

Description

DEGREES converts *angle_in_radians* to degrees.

Example

`DEGREES(PI())` equals 180.

See also

RADIANS

DOLLAR

Syntax

DOLLAR(*number*,*decimal_places*)

Description

DOLLAR formats *number* in currency style with the given number of decimal places and returns the resulting text. The *decimal_places* argument should be an integer in the interval [0,15]. If *decimal_places* is omitted, it is considered equal to two.

Example

`DOLLAR(15.25)` equals "£15.25".

`DOLLAR(15.25,0)` equals "15".

`DOLLAR(15.25,4)` equals "£15.2500".

See also

FIXED, TEXT

ERROR.TYPE

Syntax

ERROR.TYPE(*error_value*)

Description

ERROR.TYPE returns a numeric code representing the type of *error_value*, according to the following table.

If *error_value* is	ERROR.TYPE returns
#NULL!	1
#DIV/0!	2
#VALUE!	3
#REF!	4
#NAME?	5
#NUM!	6
#N/A	7
#INV!	8
#MEM!	9

ERROR.TYPE returns #N/A if *error_value* is not an error value.

Example

`ERROR.TYPE(#REF!)` equals 4.

`ERROR.TYPE(1E200*1E200)` equals 6.

See also

IS functions, TYPE

EVEN

Syntax

EVEN(*number*)

Description

EVEN rounds *number* up to the nearest even number.

Example

`EVEN(2.2)` equals 4.

`EVEN(-3.2)` equals -4.

`EVEN(2)` equals 2.

See also

ODD

EXACT

Syntax

EXACT(*text1*,*text2*)

Description

EXACT performs a case-sensitive comparison of *text1* and *text2* and returns TRUE, if they are identical, and FALSE, if they differ.

Note: use the = operator to perform a case-*insensitive* comparison.

Example

`EXACT("text","text")` equals TRUE.

`EXACT("Text","text")` equals FALSE.

See also

REPLACE, SEARCH, SUBSTITUTE

EXP

Syntax

EXP(*number*)

Description

EXP computes e^x, where e is the base of the natural logarithm.

Example

`EXP(1)` equals 2.71828182845905.

See also

LN

FACT

Syntax

FACT(*number*)

Description

FACT returns the factorial, $n!$, of *number*.

Number should be an integer value ≥ 0. If *number* is not an integer, it is truncated.

Example

`FACT(5)` equals 120.

`FACT(0)` equals 1.

See also

COMBIN, PERMUT

FALSE

Syntax

FALSE()

Description

The FALSE function returns the logical value FALSE.

FALSE takes no arguments; however, the parentheses *are* required. Note that you can also use the logical values FALSE and TRUE directly in formulas.

See also

TRUE

FIND

Syntax

FIND(*find_text*,*within_text*,*start_at_number*)

Description

FIND performs a case-sensitive search for *find_text* within *within_text*, starting at character position *start_at_number*. FIND returns the character position of the first match. Wildcard characters have no special meaning to FIND; the search is literal. The first character in *text* is numbered 1, and if you omit *start_at_number*, FIND begins the search at the beginning of *within_text*. If there is no match, FIND returns the #VALUE! error value. FIND returns #INV! if *start_at_number* is less than one or greater than the length of *within_text*. If *find_text* is the empty text, "", FIND returns 1, unless *within_text* is also the empty text, in which case it returns #INV!.

Example

`FIND("A","Agar Agar")` equals 1.

`FIND("a","Agar Agar")` equals 3.

`FIND("a","Agar Agar",4)` equals 8.

`RIGHT(A1,LEN(A1)-FIND("£",A1)+1)` equals "£100.00" if A1 contains "Payment: £100.00".

See also

SEARCH

FIXED

Syntax

FIXED(*number*,*decimal_places*,*no_commas*)

Description

FIXED converts *number* to text, with the indicated number of decimal places. If *decimal_places* is omitted, it is considered equal to two. By default, the number is formatted with commas separating thousands, but you can suppress use of commas by setting *no_commas* equal to TRUE.

Example

`FIXED(1000.25)` equals "1,000.25".

`FIXED(1000.25,4,TRUE)` equals "1000.2500".

See also

DOLLAR, TEXT

FLOOR

Syntax

FLOOR(*number*,*significance*)

Description

FLOOR rounds *number* down to the nearest multiple of *significance*. Both arguments to FLOOR must have the same sign; otherwise, FLOOR returns the #INV! error value.

Example

`FLOOR(1.25,0.2)` equals 1.2.

`FLOOR(-2.4,-0.5)` equals -2.

See also

CEILING

FRACTION

Syntax

FRACTION(*number*)

Description

FRACTION returns the fractional part of *number*. If *number* is an integer, FRACTION returns zero.

Example

`FRACTION(3.2)` equals 0.2.

`FRACTION(1.2E+100)` equals zero. Because numbers in Eureka only contain about 15 significant digits, numbers greater than 10^{15} or so can never have fractional parts.

See also

INT, TRUNC

FV

Syntax

FV(*rate*,*nper*,*pmt*,*pv*,*type*)

Description

FV returns the amount an investment is worth at the end of its lifetime. FV can compute the future value of an annuity or a single payment earning compound interest.

Rate is the periodic interest rate.

Nper is the number of periods for the annuity.

Pmt is the periodic payment of the annuity.

Pv is the initial value of the investment.

Type indicates whether payments occur at the end of periods (ordinary annuity) or at the beginning (annuity due). Set *type* to 0 to designate an ordinary annuity and 1 to indicate an annuity due. If you omit *type*, it is considered equal to 0.

For a more complete description of the arguments, see PV.

Examples

Suppose you can deposit £100 at the end of each month into an account earning 7% annual interest, compounded monthly. If you start this month, how much will this account be worth in 10 years? The formula to compute this is:

`FV(7%/12,12*10,-100)`, which equals 17308.48.

The computed future value is positive because you will receive this money, and cash inflows are considered positive in the time value of money functions. The *pmt* argument is negative, because you are paying out this amount, and cash outflows are considered negative. Thus, at the end of 10 years, you will have accumulated £17,308.48.

Suppose, however, you have £5,000 with which to start the account described above. How much will it be worth given identical interest rates and payment and compounding frequencies? The formula is:

`FV(7%/12,12*10,-100,-5000)`, which equals 27356.79.

This formula is identical to the one in the prior example, except that it defines a present value argument of -5000. The *pv* argument is negative because you are paying this money out. Thus, by making an initial deposit of £5,000 and monthly payments of £100, at the end of 10 years, you will have £27,356.79 in this account, compared to £17,308.48 if you start with an empty account.

See also

NPER, PMT, PV, RATE, IPMT, PPMT

GEOMEAN

Syntax

GEOMEAN(*number1*,*number2*,...*number14*)

Description

GEOMEAN computes the geometric mean of its arguments.

GEOMEAN accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. GEOMEAN ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, or any value ≤ 0, GEOMEAN returns the #INV! error value.

GEOMEAN is computed via the following formula:

$$GM_{\bar{y}} = \sqrt[n]{y_1 y_2 \cdots y_n}$$

Example

`GEOMEAN(2,4,3,5)` equals 3.309751.

See also

AVERAGE, HARMEAN, MEDIAN, MODE

GROWTH

Syntax

GROWTH(*y_array*,*x_array*,*new_x_values*,*const*)

Description

GROWTH returns the y-estimates for the values in *new_x_values* after calculating the coefficients for the exponential equation

$$y = bm_1^{x_1} m_2^{x_2} \ldots m_n^{x_n}$$

which is made linear in the coefficients as follows:

$$\ln y = \ln b + x_1 \ln m_1 + x_2 \ln m_2 + \ldots + x_n \ln m_n$$

GROWTH performs linear or multiple regression on this transformed equation.

Y_array should be an array or cell range containing the dependent values.

X_array should be an array or cell range containing the independent values. If you omit *x_array*, GROWTH considers it to be valued as {1, 2, 3, ..., *n*}, but in the same size and shape as *y_array*.

New_x_values can be a value, reference, or array. If you omit *new_x_values*, GROWTH considers it to be valued as {1, 2, 3, ..., *n*}, but in the same size and shape as *y_array*, unless *x_array* was specified, in which case GROWTH uses a copy of *x_array* for *new_x_values*. If *new_x_values* is an array, GROWTH returns an array of the same size and shape containing the estimates.

Const is a logical argument which controls whether GROWTH includes the constant term *b* in its calculations. If *const* is TRUE or omitted, then the constant term is included. If *const* is FALSE or 0, then TREND excludes the constant term from its calculations. . In effect, the constant is forced to equal 1 in the untransformed equation.

For a more detailed description of GROWTH's arguments, refer to LINEST.

Example

Suppose a species' major predator becomes extinct, and the population subsequently undergoes exponential growth. If censuses taken at 6 month intervals reveal populations of 10,000, 12,000, 18,000, 28,000, and 46,000, what will the population be when the next census is taken, if this trend continues unabated? The formula is:

```
GROWTH({10000,12000,18000,28000,46000},
       {0,6,12,18,24},30)
```

which equals 62650. If instead we want to predict the populations for the next 2 years, we can use:

```
{GROWTH({10000,12000,18000,28000,46000},
        {0,6,12,18,24},{30,36,42,48})}
```

which equals {62650,92528,136653,201822}. Of course, we would need to be reasonably sure that the regression equation will remain valid 2 years beyond the last recorded census, at 24 months, before taking these predictions very seriously; many factors could contribute to invalidating the computed coefficients. A few obvious ones include dwindling food supplies, epidemics, climate changes, etc.

See also

INTERCEPT, LINEST, LOGEST, PEARSON, RSQ, SLOPE, STEYX, TREND

HARMEAN

Syntax

HARMEAN(*number1*,*number2,...number14*)

Description

HARMEAN computes the harmonic mean of its arguments.

HARMEAN accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. HARMEAN ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, or any value ≤ 0, HARMEAN returns the #INV! error value.

HARMEAN is computed via the following formula:

$$\frac{1}{H_y} = \frac{1}{n}\sum\frac{1}{y}$$

Example

`HARMEAN(2,4,3,5)` equals 3.116883.

See also

AVERAGE, GEOMEAN, MEDIAN, MODE

HLOOKUP

Syntax

HLOOKUP(*lookup_value,array,row_offset*)

Description

HLOOKUP performs a "horizontal lookup" on *array*. HLOOKUP searches for *lookup_value* in the first row of *array*, and upon finding a match, returns the value in *array* in the same column as the match but in the row given by *row_offset*.

Lookup_value is the value you wish to find. It should be text, numeric, or logical. Text comparisons are case-insensitive, and wildcard characters are matched literally (that is, they do not act as wildcards).

Array is the array or cell range which HLOOKUP will search. HLOOKUP searches for *lookup_value* in the first row of *array*, and HLOOKUP expects the values in the first row to be sorted in ascending order, according to Eureka's collating scheme, in which the order is: numbers, text, logicals, error values, and blank cells.

Row_offset is a 1-based offset from the first row in *array*. That is, if *array* is the range B2:C5, then a *row_offset* of 1 refers to row 2. When HLOOKUP finds a match for *lookup_value* in the first row of *array*, it returns the value in the same column as the match, but in the row indicated by *row_offset*.

If HLOOKUP fails to find an exact match for *lookup_value*, it matches the largest value not greater than *lookup_value*. If no such value exists, HLOOKUP returns the #N/A error value. If *row_offset* is < 1, HLOOKUP returns #VALUE!, and if *row_offset* is greater than the number of rows in *array*, HLOOKUP returns the #REF! error value.

Examples

	A	B	C	D	E
1	**Floppy disk pricing**				
2		1	50	100	500
3	Generic	0.24	0.22	0.19	0.17
4	Name brand	0.48	0.43	0.39	0.35
5	Premium	0.30	0.27	0.24	0.22

The worksheet above defines an imaginary company's discounted pricing schedule for floppy disks. If we name B2:E5 Table, then:

`HLOOKUP(10,Table,2)` equals 0.24.

Lookup_value is 10, and since *row_index* is 2, we're interested in "Generic" disks. HLOOKUP compares B2:E2 to 10, and although it does not find an exact match, the "match greatest value less than or equal to *lookup_value*" rule causes it to match the entry in B2, namely 1. Thus, HLOOKUP returns the value in B3, which is the unit price for Generic disks. The entries in cells B2:E2 comprise a series of quantity ranges for discounting; column B is for quantities between 1 and 49, because in this HLOOKUP range, the entry in B2, 1, is the greatest value less than or equal to values less than 50, the entry in C2. Similarly, *lookup_values* greater than or equal to 50 but less than 100 match C2, *lookup_values* between 100 and 500 match D2, and values greater than or equal to 500 match E2.

In the example above, we've managed to index the table using a variable *lookup_value*, which allows us to select the proper price based on the quantity ordered. It would be useful to extend this, so that we could also easily vary the type of disk. What we need is a way to find the row offset for HLOOKUP, using the textual disk type. The MATCH function is perfectly suited for this, because it searches for a value in an array and returns its integer offset. If A10 contains the disk type, and B10 contains the quantity ordered, the formula is:

`HLOOKUP(B10,Table,MATCH(A10,A2:A5,0)).`

If A10 contains "Name brand" and B10 contains 100, then this formula returns 0.39, the correct value for Name brand disks, such that 100 ≤ quantity < 500. We specified 0 for MATCH's *type* argument, because we want an exact match; in this case, MATCH does *not* require its lookup area to be sorted.

As a final remark, notice that we omitted column A, which contains the disk types, from the range name Table. We had to omit this column, because A2 is blank, and including it would violate Eureka's collating order, described above, causing HLOOKUP to fail, as a blank cell would precede numeric cells in the first row.

See also

CHOOSE, LOOKUP, MATCH, VLOOKUP

HOUR

Syntax

HOUR(*serial_number*)

Description

HOUR returns the hour of the day of *serial_number*, an integer falling in the range 0-23.

You may use a textual time rather than a number; Eureka converts the text to a serial number automatically. However, text in references and arrays is *not* converted and generates a #VALUE! error when used with HOUR.

Example

`HOUR("3:30:45 PM")` equals 15.

`HOUR(33000.34)` equals 8.

See also

DAY, MONTH, YEAR, MINUTE, SECOND

IF

Syntax

IF(*test_condition*,*result_if_true*,*result_if_false*)

Description

IF returns *result_if_true* if *test_condition* evaluates to the logical value TRUE, and *result_if_false* if the condition evaluates to FALSE. For the purposes of IF, all non-zero numbers are considered equivalent to TRUE, and zero is considered FALSE.

If *result_if_true* is omitted, and *test_condition* evaluates TRUE, IF returns zero.

If *result_if_false* is omitted, and *test_condition* evaluates FALSE, IF returns FALSE.

Either or both of the result arguments can be arrays; IF will return the appropriate, entire array, provided the formula was not array-entered.

If *test_condition* is an array, and the formula was entered as an array formula, IF expands (as necessary) both result arguments into arrays, returning an array of values. If we let:

A = the array of values IF returns,

C = the *test_condition* array,

RT = the *result_if_true* array,

RF = the *result_if_false* array,

then:

$$a_{ij} = rt_{ij} \quad (c_{ij} = \text{TRUE}),$$
$$= rf_{ij} \quad (c_{ij} = \text{FALSE}).$$

This can be very useful, as it allows you to select items from two arrays, based on the contents of a third array (see the third and fourth examples below).

Example

`IF(Grade>=90,"A",IF(Grade>=80,"B",IF(Grade>=70, "C",IF(Grade>=60,"D","F"))))` assigns a letter grade based on a numeric score and scale.

`IF(NOT(ISNUMBER(A1)),"Please enter a number in A1")` equals the text "Please...", if A1 doesn't contain a number, and FALSE if it does. If you would rather this formula display nothing at all when A1 contains a number, you would specify the empty text for *result_if_false*. The formula would then be:
`IF(NOT(ISNUMBER(A1)),"Please enter a number in A1","")`.

`SUM(IF(A1:A10="Y",1,0))`, when entered as an array formula, equals the number of cells in A1:A10 that contain the text "Y," perhaps representing the word "Yes," as might be used in a survey. Note that the two result arguments can also be arrays, allowing you to select values from them based upon the condition array. Here, the numbers 1 and 0 are implicitly expanded into 10 row, 1 column arrays (to match the dimensions of A1:A10), containing all 1's and all 0's, and IF selects items from them as described above. Thus, in this formula, IF returns a 10×1 array containing only 1's and 0's to SUM, which totals them and returns a single value.

`AVERAGE(IF(A1:A10<>0,A1:A10,""))`, when entered as an array formula, computes the average of the values in A1:A10, *excluding* zero values. Note the use of the empty text as the *result_if_false* argument. It is expanded into a 10×1 array, and IF selects items from it when the corresponding elements in A1:A10 are zero. But since AVERAGE ignores text inside arrays and references, this also means AVERAGE ignores zero values.

`MAX(IF(A1,B1:B10,C1:C10))` equals the maximum value in B1:B10, if A1 evaluates to TRUE, and C1:C10 if A1 evaluates FALSE. Note that since the condition is based on a single value, IF simply returns the appropriate result argument, a range in this case, and there is no need to enter this as an array formula, because MAX operates on ranges as well as single cells.

See also

AND, OR, NOT

INDEX

Syntax

INDEX(*reference*,*row_offset,column_offset,area_num***)**

INDEX(*array*,*row_offset,column_offset***)**

Description

In its reference form, INDEX returns the reference of the cell within *reference* at the intersection of *row_offset* and *column_offset*, which are 1-based indexes, measured from the cell at the upper left corner of *reference*. If *reference* is a multiple reference, then *area_num* is a 1-based index into the multiple reference, and it selects a single reference for INDEX to use.

In its array form, INDEX returns the value within *array* at the intersection of *row_offset* and *column_offset*, which again are 1-based indexes into *array*, measured from the upper left corner of the array.

The two forms of INDEX are quite similar, and to simplify the discussion, we will define *matrix* to mean the first argument to INDEX, which can be either a reference or array. (If the first argument is a multiple reference, then *matrix* refers to the reference selected by *area_num*.)

You must specify at least one of *row_offset*, *column_offset*, and *area_num*. *Row_offset* should range from 1 to the number of rows in *matrix*, and *column_offset* should range from 1 to the number of columns in *matrix*. *Area_num* should range from 1 to the number of references in the multiple reference.

If you specify a non-zero *row_offset* but set *column_offset* to zero, then INDEX returns the entire row indicated by *row_offset*. This will be a reference, such as A1:F1 for the reference form of INDEX, and a vector, such as {1,2,3}, for the array form of INDEX.

If you specify a non-zero *column_offset* but set *row_offset* to zero, then INDEX returns the entire column indicated by *column_offset*. This will be a reference, such as A1:A5 for the reference form of INDEX, and a vector, such as {1;2;3}, for the array form of INDEX.

If you set both *row_offset* and *column_offset* to zero, then INDEX simply returns *matrix*.

If *matrix* is a vector, that is, a reference or array with only one row or column, then you can omit one of the *offset* arguments, as follows. If *matrix* contains only one column, then you can omit *column_offset*. If *matrix* contains only one row, then you can omit *row_offset*.

Examples

Consider the following worksheet, which presents hypothetical sales information for some imaginary recording. We'll name A1:D3 Analogue, A5:D6 Digital, and A1:D6 Recording.

	A	B	C	D
1		Unit Price	# Sold	Total Sales
2	Albums	£5.00	4	£20.00
3	Cassettes	£5.50	15	£82.50
4				
5	Compact disks	£8.00	24	£192.00
6	Digital Audio Tape	£14.00	3	£42.00

`INDEX(Recording,5,4)` equals the reference D5, containing £192.00, the total sales of compact disks.

`INDEX(Digital,1,4)` also equals the reference D5, containing £192.00.

`SUM(INDEX(Analogue),0,4)` equals `SUM(D1:D3)` equals £102.50, the total sales for analogue recordings.

`SUM(INDEX((Analogue,Digital),0,4,2))` equals
`SUM(INDEX((A1:D3,A5:D6),0,4,2))` equals
`SUM(INDEX(A5:D6,0,4))` equals `SUM(D5:D6)`, the total sales of digital recordings, £234.00.

`SUM(D1:INDEX(Digital,1,4))` equals `SUM(D1:D5)` equals the total sales in all formats except DAT, £294.50.

Now, consider an array example.

`INDEX({1,2,3;4,5,6},1,2)` equals 2.

`INDEX(LINEST(Y,X),1,2)` returns the intercept in a simple linear regression.

See also

CHOOSE, HLOOKUP, LOOKUP, MATCH, VLOOKUP

INT

Syntax

INT(*number*)

Description

INT converts *number* to an integer, returning the greatest integer not greater than *number*. Thus, if number contains a fractional part, INT truncates it and returns the next smallest integer. However, if number is negative, the next smallest integer will be one less in magnitude than the actual integer part of number. Use TRUNC to get the true integer part, without this "rounding- down" effect.

Example

`INT(2.3)` equals 2.

`INT(-2.3)` equals -3.

`INT(4)` equals 4.

`INT(-4)` equals -4.

See also

MOD, ROUND, TRUNC

INTERCEPT

Syntax

INTERCEPT(*y_array,x_array*)

Description

INTERCEPT computes the value a in the linear regression $y = a+bx$.

Y_array and *x_array* should be arrays or references with identical dimensions. INTERCEPT ignores blank cells and text and logical values within references and arrays. A number occurring in one array must be matched by a number at the same position in the complementary array; otherwise, INTERCEPT returns the #VALUE! error value. INTERCEPT returns the #INV! error value if both *y_array* and *x_array* are empty.

The formula for INTERCEPT is:

$$a = \overline{y} - b\overline{x}$$

where b is determined as in SLOPE.

Example

`INTERCEPT({1.2,2.5,3.3,4.6},{1,2,3,4})` equals 0.15.

See also

LINEST, SLOPE, TREND

IPMT

Syntax

IPMT(*rate,per,nper,pv,*fv,type)

Description

IPMT returns the amount of the interest paid during a given period of an amortization schedule.

Rate is the periodic interest rate.

Per is the period for which you wish to determine the interest paid. It must range from 1 to *nper*.

Nper is the number of periods for the annuity.

Pv is the present value of the annuity, the amount it is worth today.

Fv is the future value of the annuity, the amount it will be worth after the last period.

Type indicates whether payments occur at the end of periods (ordinary annuity) or at the beginning (annuity due). Set *type* to 0 to designate an ordinary annuity and 1 to indicate an annuity due. If you omit *type*, it is considered equal to 0.

For a more complete description of the arguments, see PV.

Examples

Suppose you take out a four year car loan for £15,000. The annual interest rate is 11%, and interest is compounded monthly. How much of the first payment goes to interest? The formula to compute this is:

`IPMT(11%/12,1,48,15000)`, which equals -137.50.

Thus, £137.50 of the first payment goes to interest.

See also

PPMT, FV, NPER, PMT, PV, RATE

IQRANGE

Syntax

IQRANGE(*array*)

Description

IQRANGE returns the inter-quartile range, the difference between the 3rd and 1st quartiles. Thus, IQRANGE(a) is equivalent to QUARTILE(a,3)-QUARTILE(a,1).

Example

`IQRANGE({1,2,3,4,5,6,7})` equals 4.

See also

LARGE, MAX, MEDIAN, MIN, QUARTILE, SMALL

IRR

Syntax

IRR(*values*,*guess*)

Description

IRR computes the internal rate of return, the interest rate that results in a net present value of 0.

Values is a cell range, multiple area, or array containing at least one positive and one negative number representing the cash flows. Usually, the first value will be negative, representing the initial cost of the investment. IRR ignores blank cells and text and logical values within references and arrays. Be aware that the order of the values determines the order of the cash flows and influences the value of IRR.

Guess, if specified, should be reasonably close to the internal rate of return. If omitted, IRR consider *guess* equal to 0.1 or 10%. IRR uses an iterative process to determine the internal rate of return; if the absolute value of the difference between successive approximations drops below 10^{-7}, IRR terminates successfully. If this does not happen within 20 iterations, IRR returns the #NUM! error value. If this should occur, first make sure *values* contains at least one sign change, and if the contents of *values* appear proper, try adjusting *guess*. IRR will usually conclude successfully with a *guess* between 0 and 1.

Example

Suppose you are considering an investment which requires you to pay 30,000 up front, and you anticipate cash flows of 7,000, 8,000, 10,000, and 12,000 at the end of each of the next 4 years. The formula to compute the internal rate of return is:

`IRR({-30000,7000,8000,10000,12000})` equals 0.08135 or 8.135%. This means that in addition to recovering the initial outlay, the investment will earn 8.135% annual interest.

See also

MIRR, NPV

IS

Syntax

IS(*value*)

Description

The IS functions allow you to test whether *value* falls into a certain class of values, returning TRUE if it does and FALSE if it does not.

Function	Returns TRUE if
ISBLANK	Value is a blank cell.
ISERR	Value is an error value other than #N/A.
ISERROR	Value is any error value, including #N/A.
ISLOGICAL	Value is TRUE or FALSE.
ISNA	Value equals #N/A.
ISNONTEXT	Value is not text.
ISNUMBER	Value is a number.
ISREF	Value is a reference.
ISTEXT	Value is text.

Unlike most other worksheet functions, arguments to the IS family are not coerced to values or text; rather, they are used as is.

Examples

`ISREF(A1)` equals TRUE.

`ISTEXT(A1)` equals FALSE if A1 contains 1.

`ISNUMBER(A1)` equals TRUE if A1 contains 1.

See also

TYPE, ERROR.TYPE

LARGE

Syntax

LARGE(*array,k*)

Description

LARGE returns the k^{th} largest value occurring in *array*.

Array should be an array or reference containing numbers. LARGE ignores blank cells and text and logical values within arrays and references.

K should be a value ranging from 1 to the number of values in *array*. To obtain the largest value, set *k* to 1, to obtain the second largest value, set *k* to 2, and so on.

If *array* is empty, or *k* is < 1 or exceeds the number of values in *array*, LARGE returns the #INV! error value.

Example

`LARGE({2,1,5,4},2)` equals 4.

See also

MAX, MEDIAN, MIN, QUARTILE, SMALL

LEFT

Syntax

LEFT(*text*,*number_of_characters*)

Description

LEFT returns the left-most *number_of_characters* from *text*.

If *number_of_characters* is omitted, LEFT returns the first character in *text*. If *number_of_characters* exceeds the length of *text*, LEFT returns *text*.

Example

`LEFT("1230 AnyStreet",4)` equals "1230".

See also

MID, RIGHT

LEN

Syntax

LEN(*text*)

Description

LEN returns the number of characters in *text*.

Cell text is limited to 255 characters, so LEN returns a number in the range 0-255.

Example

`LEN("This is some text.")` equals 18.

`LEN("")` equals 0.

See also

EXACT, SEARCH

LINEST

Syntax

LINEST(*y_array*,*x_array,const,stats***)**

Description

LINEST computes the multiple regression $y=b+m_1x_1+m_2x_2+\ldots+m_nx_n$. In its simplest form, LINEST computes the linear regression $y=b+mx$. In addition to returning all the coefficients of the regression equation as an array, which has the form $\{m_n, \ldots, m_2, m_1, b\}$. LINEST can also include various additional regression statistics in the array it returns, and it can exclude the constant term b from its calculations.

Y_array should be an array or cell range containing the dependent values. If *y_array* is a single column, then *x_array* can contain multiple columns, each of which represents a single independent variable, ordered $x_1, x_2, \ldots x_n$ from left to right. In this case, *x_array* should contain exactly as many rows as *y_array*. If *y_array* is a single row, then *x_array* can contain multiple rows, each of which represents a single independent variable, ordered $x_1, x_2, \ldots x_n$ from top to bottom. In this case, *x_array* should contain exactly as many columns as *y_array*. Finally, if *y_array* has more than one row or column, LINEST performs a simple linear regression, and *x_array* should be exactly the same size and shape as *y_array*.

X_array should be an array or cell range containing the independent values; rules governing its shape are described above.

If you omit *x_array*, LINEST considers it to be valued as {1, 2, 3, ..., *n*}, but in the same size and shape as *y_array*.

Const is a logical argument which controls whether LINEST includes the constant term *b* in its calculations. If *const* is TRUE or omitted, then the constant term is included. If *const* is FALSE or 0, then LINEST excludes the constant term from its calculations (this is the *zero* or *no-intercept model*).

Stats is a logical argument which controls whether LINEST returns additional regression statistics. If *stats* is FALSE or omitted, LINEST returns a horizontal array containing only the regression coefficients, in the form $\{m_n, \ldots, m_2, m_1, b\}$. If *stats* equals TRUE, then LINEST returns the following additional statistics:

Statistic	**Description**
se_{coeff}	Standard errors of the regression equation coefficients, where se_1 corresponds to the estimate for m_1, se_2 to m_2, and so on through se_n. The standard error se_b represents the standard error of the intercept.
r^2	The coefficient of determination.
se_y	Standard error of the *y*-estimate.
F	The F statistic.
df	The F statistic denominator degrees of freedom.
SSR	The regression sum of squares.
SSE	The error sum of squares.

When *stats* is TRUE, the returned array has the form:

m_n	m_{n-1}	...	m_2	m_1	b
se_n	se_{n-1}	...	se_2	se_1	se_b
r^2	se_y				
F	df				
SSR	SSE				

LINEST populates all the blank entries in this array with #N/A error values, which for clarity, are not shown here.

Be aware that when *const* is FALSE, LINEST returns the *uncorrected* sums of squares.

Examples

Suppose we have a substance in solution, and we know it follows the Lambert-Beer law of light absorption. If we have several solutions of known concentration, we can use LINEST to find the regression line, which will let us determine the concentration of the substance in an unknown. If the concentrations in the known solutions are 5, 10, 15, and 20, and the absorbance values are 0.11, 0.21, 0.34, and 0.43, then the formula is:

```
{LINEST({5,10,15,20},{0.11,0.21,0.34,0.43})}
```

which equals {45.66, 0.057}. Thus the slope of the regression line is 45.66, and the y-intercept is 0.057, which is reasonably close to the zero value required by Beer's law. If the absorbance of an unknown solution is 0.28, then we can find the concentration of the substance by:

- `45.66*0.28+0.057`,
- `SUM(LINEST({5,10,15,20},{0.11,0.21,0.34,0.43})*{0.28,1})` which evaluates to:

  `SUM({45.66,0.057}*{0.28,1})`,
- `TREND({5,10,15,20},{0.11,0.21,0.34,0.43},0.28)`.

All the above formulas return a concentration of 12.84.

Suppose we have a random sample of apartment rental rates, along with their sizes and number of bedrooms and bathrooms.

	A	B	C	D
1	Rate (Y)	Size (X1)	Bedrooms (X2)	Bathrooms (X3)
2	£290	1050	1	1
3	£280	1000	1	1.5
4	£300	1050	2	1
5	£350	1250	2	1.5
6	£325	1200	1	1
7	£400	1300	2	2
8	£420	1400	2	2
9	£390	1250	2	2
10	£400	1250	2	1.5
11	£450	1500	2	2

The regression formula is:

```
{LINEST(A2:A11,B2:D11,,1),
```

which results in the following output:

	A	B	C	D
12	25.9256	20.24358	0.2686448	-43.18866
13	18.02929	13.95258	0.0503848	43.113677
14	0.956221	15.28564	#N/A	#N/A
15	43.68429	6	#N/A	#N/A
16	30620.6	1401.904	#N/A	#N/A

Thus, the regression equation is:

$y = -43.19 + 0.269x_1 + 20.24x_2 + 25.93x_3$.

See also

GROWTH, INTERCEPT, LOGEST, PEARSON, RSQ, SLOPE, STEYX, TREND

LN

Syntax

LN(*number*)

Description

LN returns the base *e*, or natural, logarithm of *number*. LN returns #INV! if *number* is zero or negative.

Example

`LN(EXP(1))` equals 1.

`LN(10)` equals 2.302585.

See also

EXP, LOG, LOG10

LOG

Syntax

LOG(*number*,*base***)**

Description

LOG computes the logarithm of *number*, allowing you to specify the base.

Base is an optional argument; if it is omitted, it is given the value 10, in which case LOG is equivalent to LOG10.

LOG returns #INV! if *number* or *base* are zero or negative.

Example

`LOG(8,2)` equals 3.

`LOG(10)` equals LOG10(10) equals 1.

See also

LOG10, EXP, LN

LOG10

Syntax

LOG10(*number*)

Description

LOG10 computes the base-10 logarithm of *number*. LOG10 returns #INV! if *number* is zero or negative.

Example

`LOG10(10)` equals 1.

`LOG10(10^9)` equals 9.

See also

LOG, EXP, LN

LOGEST

Syntax

LOGEST(*y_array*,x_array*,const,stats*)

Description

LOGEST fits the exponential curve described by:

$$y = bm_1^{x_1} m_2^{x_2} \ldots m_n^{x_n}$$

which is made linear in the coefficients as follows:

$$\ln y = \ln b + x_1 \ln m_1 + x_2 \ln m_2 + \ldots + x_n \ln m_n$$

LOGEST performs linear or multiple regression on this transformed equation, and returns the coefficients of the regression equation as an array, which has the form $\{m_n, \ldots, m_2, m_1, b\}$. Although LOGEST computes the logarithms of the coefficients, when building the array it restores each coefficient to its untransformed state. Like LINEST, LOGEST can return additional regression statistics, and it can also include various additional regression statistics in the array it returns, and it can exclude the constant term b from its calculations.

Y_array should be an array or cell range containing the dependent values.

X_array should be an array or cell range containing the independent values.

Const is a logical argument which controls whether LOGEST includes the constant term b in its calculations. If *const* is TRUE or omitted, then the constant term is included. If *const* is FALSE or 0, then LOGEST excludes the constant term from its calculations. In effect, the constant is forced to equal 1 in the untransformed equation.

Stats is a logical argument which controls whether LOGEST returns additional regression statistics. If *stats* is FALSE or omitted, LOGEST returns a horizontal array containing only the regression coefficients, in the form $\{m_n, \ldots, m_2, m_1, b\}$. If *stats* equals TRUE, then LOGEST returns several additional statistics.

See LINEST for more detailed descriptions of LOGEST arguments and the structure of the returned array. Everything stated about LINEST applies equally well to LOGEST, because LOGEST effectively calls LINEST after transforming the exponential equation, as described above. However, one significant difference concerns the additional statistics returned when *stats* is TRUE. Although LOGEST does return untransformed coefficient estimates, all the additional statistics are based on the *transformed* linear equation.

Example

Suppose a species' major predator becomes extinct, and the population subsequently undergoes exponential growth. If censuses taken at 6 month intervals reveal populations of 10,000, 12,000, 18,000, 28,000, and 46,000, the formula is:

```
{LOGEST({10000,12000,18000,28000,46000},
         {0,6,12,18,24})}
```

which equals {1.0671,8916.11}, corresponding to the equation:

$$y = 8916.11 * 1.0671^{x}$$

where x represents the elapsed time in months.

If we set *stats* to TRUE, then LOGEST returns:

	A	B
1	1.067149	8916.11
2	0.005692	0.083654
3	0.977506	0.107997
4	130.368	3
5	1.52054	0.03499

See also

GROWTH, INTERCEPT, LINEST, PEARSON, RSQ, SLOPE, STEYX, TREND

LOOKUP

Syntax

LOOKUP(*lookup_value,lookup_vector,result_vector*)

Description

LOOKUP searches for *lookup_value* in *lookup_vector* and upon finding a match, returns the corresponding value in *result_vector*.

Lookup_value is the value you wish to find. It should be text, numeric, or logical. Text comparisons are case-insensitive, and wildcard characters are matched literally (that is, they do not act as wildcards).

Lookup_vector is the one-dimensional array or cell range which LOOKUP will search. LOOKUP expects the values in the *lookup_vector* to be sorted in ascending order, according to Eureka's collating scheme, in which the order is: numbers, text, logicals, error values, and blank cells.

Result_vector is the one-dimensional array or cell range containing the result values. LOOKUP extracts its return value from this array by choosing the element offset from the first element by the same amount that *lookup_value* is offset from the first element of *lookup_vector*. *Result_vector* and *lookup_vector* thus do not need to have the same shape.

If LOOKUP fails to find an exact match for *lookup_value*, it matches the largest value not greater than *lookup_value*. If no such value exists, LOOKUP returns the #N/A error value. If there is no element in *result_vector* complementary to the matched element in *lookup_vector*, as will occur if the 10th element in *lookup_vector* is matched but *result_vector* contains only 5 elements, LOOKUP returns the #REF! error value.

LOOKUP is very similar to HLOOKUP and VLOOKUP; in fact, it provides a super-set of their functionalities. LOOKUP has the additional capability to perform a horizontal search, yet return a value from a vertical array, and vice versa, and unlike HLOOKUP and VLOOKUP, cell ranges used for *lookup_vector* and *result_vector* do not need to begin in the same column or same row.

Examples

	A	B	C	D	E
1	**Floppy disk pricing**				
2		1-49	50-99	100-499	500+
3	Generic	0.24	0.22	0.19	0.17
4	Name brand	0.48	0.43	0.39	0.35
5	Premium	0.30	0.27	0.24	0.22

The worksheet above defines an imaginary company's discounted pricing schedule for floppy disks.

`LOOKUP("Premium",A3:A5,C3:C5)` equals 0.27, the unit price for Premium disks where 50 ≤ quantity < 100.

If A10 contains the disk type, and B10 contains the quantity ordered, then:

```
LOOKUP(A10,A3:A5,CHOOSE((B10>=1)+(B10>=50)+
(B10>=100)+(B10>=500),B3:B5,C3:C5,D3:D5,E3:E5))
```

extracts the proper unit price based on both the disk type and the quantity ordered, and you easily vary these two index cells, A10 and B10, for different orders. For example, if A10 contains "Generic" and B10 contains 100, the formula is evaluated to:

`LOOKUP("Generic",A3:A5,CHOOSE(1+1+1+0,B3:B5, C3:C5,D3:D5,E3:E5)),` which in turn becomes:

`LOOKUP("Generic",A3:A5,D3:D5)` equals 0.19, the proper unit price for Generic disks where 100 ≤ quantity < 500. The CHOOSE function selects the proper *result_vector* by using *logical switch* expressions. A logical switch is an expression such as B10>=50, which equals either TRUE or FALSE. When TRUE or FALSE are used in expressions, Eureka translates them to 1 and 0, respectively, allowing us to easily compute the offset into the CHOOSE argument list.

See also

CHOOSE, HLOOKUP, MATCH, VLOOKUP

LOWER

Syntax

LOWER(*text*)

Description

LOWER converts all upper-case characters in *text* to lower-case and returns the modified text.

Example

`LOWER("A.B.c.d.")` equals "a.b.c.d."

See also

UPPER

MATCH

Syntax

MATCH(*lookup_value,lookup_vector,type*)

Description

MATCH returns the position of *lookup_value* in *lookup_vector*.

Lookup_value is the value you wish to find. It should be text, numeric, or logical. Text comparisons are case-insensitive, and wildcard characters are matched literally (that is, they do not act as wildcards), unless *type* is zero.

Lookup_vector is the one-dimensional array or cell range which MATCH will search.

Type controls how MATCH performs the comparisons between the *lookup_value* and the values in *lookup_vector*.

- If *type* is 1, then MATCH returns the position of the largest value less than or equal to *lookup_value;* this is also the search method used by HLOOKUP, LOOKUP, and VLOOKUP. When *type* is 1, MATCH expects the values in the *lookup_vector* to be sorted in ascending order, according to Eureka's collating scheme, in which the order is: numbers, text, logicals, error values, and blank cells. If there are multiple matching elements in *lookup_vector*, MATCH returns the position of the *last* matching element.
- If *type* is 0, then MATCH performs an exact match, although wildcards are permitted in text searches when *type* equals 0. Unlike *type* values of -1 and 1, a value of 0 does not require that *lookup_vector* be sorted, and rather than returning the position of the last element in a set of multiple

matches, MATCH returns the position of the *first* matching element in *lookup_vector*, when *type* equals 0.

- If *type* is -1, then MATCH returns the position of the smallest value greater than or equal to *lookup_value*. When *type* is -1, MATCH expects the values in the *lookup_vector* to be sorted in descending order, according to Eureka's collating scheme, in which the order is: error values, logicals, text, numbers, and blank cells. If there are multiple matching elements in *lookup_vector*, MATCH returns the position of the *last* matching element.

If you omit *type*, MATCH considers it equal to 1.

If MATCH fails to find a match, it returns the #N/A error value.

Examples

	A	B	C	D	E
1	**Floppy disk pricing**				
2		1	50	100	500
3	Generic	0.24	0.22	0.19	0.17
4	Name brand	0.48	0.43	0.39	0.35
5	Premium	0.30	0.27	0.24	0.22

The worksheet above defines an imaginary company's discounted pricing schedule for floppy disks.

`MATCH("Premium",A3:A5,0)` equals 3, the position of Premium within the range A3:A5.

If A10 contains the disk type, and B10 contains the quantity ordered, then we can use MATCH in combination with INDEX to easily look up the unit price based on these two parameters. The formula is:

```
INDEX(B3:E5,MATCH(A10,A3:A5,0),MATCH(B10,B2:E2))
```

Notice that for INDEX's *row_index* argument, we use a MATCH *type* of 0, requesting an exact match, while for the *column_index* argument, we omit *type*, and as described above, MATCH considers it equal to 1. Thus, if A10 contains "Generic" and B10 contains 150, the formula evaluates to `INDEX(B3:E5,1,3)`, which equals 0.19, the correct unit price for Generic disks such that $100 \leq$ quantity < 500.

See also

CHOOSE, HLOOKUP, LOOKUP, VLOOKUP

MAX

Syntax

MAX(***number1***,*number2*,...*number14*)

Description

MAX returns the maximum value occurring within its arguments.

MAX accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. MAX ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, MAX returns zero.

Example

`MAX(2.4,3.2,2.8)` equals 3.2.

`MAX(A1:A3)` equals 3.2 if A1:A3 contains the values 2.4, 3.2, and 2.8.

See also

MIN

MDETERM

Syntax

MDETERM(*array*)

Description

MDETERM returns the determinant of the square matrix *array*. A matrix is invertible if and only if its determinant is not equal to zero.

If *array* is not square, MDETERM returns the error value #INV!.

Example

`MDETERM({1,2;3,4})` equals -2.

`MDETERM({1,2;2,4})` equals 0.

See also

MINVERSE, MMULT

MEDIAN

Syntax

MEDIAN(*number1*,*number2*,...*number14*)

Description

MEDIAN returns the median value of its arguments. If *k* values are supplied, then for odd *k*, MEDIAN returns the value at position (*k*+1)/2 in the sorted list of numbers. For even *k*, MEDIAN returns the average of the values at positions *k*/2 and *k*/2+1.

MEDIAN accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. MEDIAN ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, MEDIAN returns the #INV! error value.

Example

`MEDIAN(2.4,3.2,2.8)` equals 2.8.

`MEDIAN(1,2,3,4)` equals 2.5.

See also

AVERAGE

MID

Syntax

MID(*text,start_num,num_chars*)

Description

MID extracts a portion of *text* beginning at character position *start_num* with length *num_chars*. The first character in *text* is numbered 1. If *start_num* is less than 1, MID returns the #INV! error value. If *start_num* exceeds the length of *text*, MID returns the empty text, "". If *num_chars* is less than zero, MID returns #INV!, while if *num_chars* equals zero, MID returns the empty text. Finally, *start_num* + *num_chars* may exceed the length of *text* without error.

Example

`MID("First Middle Last",7,6)` equals "Middle".

See also

LEFT, RIGHT

MIN

Syntax

MIN(*number1*,*number2*,...*number14*)

Description

MIN returns the minimum value occurring within its arguments.

MIN accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. MIN ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, MIN returns zero.

Example

`MIN(2.4,3.2,2.8)` equals 2.4.

`MIN(A1:A3)` equals 2.4 if A1:A3 contains the values 2.4, 3.2, and 2.8.

See also

MAX

MINUTE

Syntax

MINUTE(*serial_number*)

Description

MINUTE returns the minute of *serial_number*, an integer falling in the range 0-59.

You may use a textual time rather than a number; Eureka converts the text to a serial number automatically. However, text in references and arrays is *not* converted and generates a #VALUE! error when used with MINUTE.

Example

`MINUTE("3:30:45 PM")` equals 30.

`MINUTE(33000.34)` equals 9.

See also

DAY, MONTH, YEAR, HOUR, SECOND

MINVERSE

Syntax

MINVERSE(*array*)

Description

MINVERSE returns the inverse of the square matrix *array*, which can be used to find the solution to a set of linear equations. The matrix product of a matrix and its inverse is the identity matrix; however, due to limited precision in computing the inverse, this product may not exactly equal the identity matrix. When this occurs, typically some elements off the diagonal in the product $\mathbf{AA}^{-1}$ will be very small (10^{-14} or thereabouts) but not zero.

If *array* is square but not invertible, MINVERSE returns the error value #NUM!. If *array* is not square, MINVERSE returns the error value #INV!.

Example

`MINVERSE({1,2;3,4})` equals {-2,1;1.5,-0.5}.

`MMULT({1,2;3,4},MINVERSE({1,2;3,4})` equals {1,0;0,1}.

`MINVERSE({1,2;2,4})` equals {#NUM!,#NUM!;#NUM!,#NUM!}.

See also

MDETERM, MMULT

MIRR

Syntax

MIRR(*values,finance_rate,reinvest_rate*)

Description

MIRR computes the modified internal rate of return.

Values is a cell range, multiple area, or array containing the cash flows, including at least one positive and one negative number. MIRR ignores blank cells and text and logical values within references and arrays. Be aware that the order of the values determines the order of the cash flows and influences the value of MIRR.

Finance_rate is the interest rate you are charged on money you borrow to pay for the investment.

Reinvest_rate is the interest rate you receive as you reinvest the cash inflows.

The formula for MIRR is:

$$\mathrm{MIRR} = \left(\frac{-\mathrm{NPV}(rrate\ ,\ values[positive])\ *\ (1 + rrate)^{n}}{\mathrm{NPV}(frate\ ,\ values[negative])*(1 + frate)} \right)^{\frac{1}{n-1}} - 1$$

where n equals the number of cash flows, *rrate* equals *reinvest_rate*, and *frate* equals *finance_rate*.

Example

`MIRR({-80000,20000,40000,-10000,50000},12%,8%)`
equals 0.0875 or 8.75%.

See also

IRR, NPV

MMULT

Syntax

MMULT(*array_mn,array_np*)

Description

MMULT multiplies *array_mn*, an array with m rows and n columns, by *array_np*, an array with n rows and p columns, and returns the matrix product, an array with m rows and p columns. If a represents an item in the matrix product, b an item in *array_mn*, and c an item in *array_np*, then the formula used is:

$$a_{ij} = \sum_{k=1}^{n} b_{ik} c_{kj}$$

Example

`MMULT({1,2,3},{2;3;4})` equals 20.

`MMULT({1,2,3;4,5,6},{7;8;9})` equals {50;122}.

See also

MDETERM, MINVERSE

MOD

Syntax

MOD(*number,divisor*)

Description

MOD returns the remainder of *number* ÷ *divisor*. If *divisor* is zero, MOD returns the #DIV/0! error value. MOD is related to INT by the equation:

MOD(n,d) = n-d*INT(n/d)

Example

`MOD(5,3)` equals 2.

`MOD(-5,3)` equals 1.

`MOD(5,-3)` equals -1.

`MOD(-5,-3)` equals -2.

See also

INT, TRUNC

MODE

Syntax

MODE(*number1*,*number2,...number14*)

Description

MODE returns the most frequently occuring value within its arguments.

MODE accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. MODE ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If the mode does not exist, MODE returns the #N/A error value.

Example

`MODE(1,2,2,1,3,1)` equals 1.

See also

AVERAGE, MEDIAN

MONTH

Syntax

MONTH(*serial_number*)

Description

MONTH returns the month of *serial_number*, an integer falling in the range 1-12.

You may use a textual date rather than a number; Eureka converts the text to a serial number automatically. However, text in references and arrays is *not* converted and generates a #VALUE! error when used with MONTH.

Example

`MONTH("22-Sep-92")` equals 9.

`MONTH(33000.34)` equals 5.

See also

DAY, YEAR, HOUR, MINUTE, SECOND

N

Syntax

N(*value*)

Description

N returns *value* converted to a number.

If *value* is or refers to a number, N returns that number. If *value* is the logical value TRUE, N returns 1. Otherwise, N returns 0.

Example

`N(A1)` equals 2 if A1 contains the number 2.

`N(TRUE)` equals 1.

`N("22-Sep-92")` equals 0, because *value* is text.

See also

T

NA

Syntax

NA()

Description

NA returns the error value #N/A.

NA takes no arguments; however, the parentheses *are* required. Note that you can also use #N/A and all other error values directly in formulas.

Example

`IF(A1<>0,A1,NA())` equals #N/A if A1 contains zero or is blank.

See also

ISNA

NOT

Syntax

NOT(*logical*)

Description

NOT returns TRUE if *logical* evaluates FALSE and FALSE if *logical* evaluates TRUE.

NOT ignores blank cells and text within references and arrays; however, for the purposes of NOT, non-zero numbers are considered equivalent to TRUE, and zero is considered equivalent to FALSE. A reference argument may be a range as well as a single cell. If no logicals are supplied, NOT returns the #VALUE! error value.

Example

`NOT(TRUE)` equals FALSE.

`NOT(1>2)` equals `NOT(FALSE)` equals TRUE.

See also

AND, OR, IF

NOW

Syntax

NOW()

Description

NOW returns the serial number representing today's date and the current time, according to your computer's clock.

Unlike TODAY, NOW returns a real number including a fractional time component. NOW is a *volatile* function; that is, formulas using NOW are recalculated upon data entry and are included in all other recalculations. In addition, when you load a file, Eureka recalculates all formulas using NOW.

NOW takes no arguments; however, the parentheses *are* required.

Example

`NOW()` equals 33871.770833, if today's date is 24-Sep-92, and the time is 6:30 PM.

See also

TODAY

NPER

Syntax

NPER(*rate*,*pmt*,*pv*,*fv*,*type*)

Description

NPER returns the number of periods for an annuity or compound interest on a lump sum.

Rate is the periodic interest rate.

Pmt is the periodic payment for an annuity.

Pv is the present value of the investment.

Fv is the future value of the investment.

Type indicates whether payments occur at the end of periods (ordinary annuity) or at the beginning (annuity due). Set *type* to 0 to designate an ordinary annuity and 1 to indicate an annuity due. If you omit *type*, it is considered equal to 0.

For a more complete description of the arguments, see PV.

Examples

Suppose you take out a loan for £10,000. The annual interest rate is 11%, and interest is compounded monthly. If your monthly payment is to be £200, how many months will it take you to pay off the loan? The formula to compute this is:

`NPER(11%/12,-200,10000)`, which equals 67.19.

Thus, it will take you 67.19 months to pay off the loan; the fractional part indicates that after the 67th payment, you will have some amount left to pay which is less than the monthly payment of £200. The *pmt* argument is negative, because you will be paying this amount, and cash outflows are negative. The *pv* argument is positive, since you initially receive this amount. Finally, the *fv* argument is omitted and thus zero, which is appropriate, since the loan is to be fully paid off.

Suppose instead you have £10,000 to deposit in an account today, and you want to determine how long it will take to double to £20,000, based on an account drawing 7% annual interest, compounded monthly, with no interim deposits or withdrawals. This is not an annuity calculation but rather a compound interest calculation on a lump sum. The formula to compute the number of periods is:

`NPER(7%/12,,-10000,20000)`, which equals 119.17.

Thus, it will take almost 10 years for the initial deposit to double to £20,000 in an account drawing 7% annual interest, compounded monthly. Notice that since this is not an annuity calculation, we omit the *pmt* argument; NPER considers it equal to zero. The *pv* argument is negative, since you are initially depositing this money to an account, which is a cash outflow. The *fv* argument is positive, because at the end of the investment, you will receive this amount, and cash inflows are always positive.

See also

FV, PMT, PV, RATE, IPMT, PPMT

NPV

Syntax

NPV(*rate*,*number1*,*number2*,...*number13*)

Description

NPV computes the net present value of a stream of equally-spaced cash flows at a given discount or hurdle rate. Unlike the PV function, NPV allows for a series of unequal payments.

Rate is the hurdle rate over a period.

The remaining *number* arguments should be numbers, or references or arrays containing numbers. NPV ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. Be aware that the order of the *number* arguments determines the order of the cash flows and influences the value of NPV.

NPV is computed via the following formula, where n is the number of payments:

$$\mathrm{NPV} = \sum_{i=1}^{n} \frac{value_i}{(1 + rate)^i}$$

This formula assumes that payments occur at the ends of periods. If all the cash flows are positive, then NPV returns the total amount you would need today in order to create an account earning the hurdle rate, such that you could withdraw $value_i$ at the end of each period i. If there is an initial cost associated with the investment, then the difference NPV-cost is a measure of its acceptability. If this difference is > 0, then the investment can be considered acceptable, because the investment is "worth more" than its cost at the indicated hurdle rate (*hurdle*, because the rate must exceed or "hurdle" some value so that NPV-cost ≥ 0, Only when this condition is satisfied can the investment be considered to be an acceptable one).

Finally, be aware that you should include the initial cost in the *number* arguments only if it occurs at the end of the first period. If instead it occurs at the beginning of the first period, as it commonly does, the true net present value is given by NPV-cost.

Examples

Suppose you have the chance to invest £10,000 in an enterprise, and you expect the annual income for this investment to be £2,500, £5,000, and £7,500 over the next 3 years, and the hurdle rate is 8%. The formula is:

`NPV(8%,2500,5000,7500)`, which equals 12555.25.

Since the NPV result exceeds the initial cost, you can consider this an acceptable investment.

Suppose instead you have the investment opportunity described above, except that you expect it to lose £2,000 the first year (the lifetime, then, is 4 years). The formula is:

`NPV(8%,-2000,2500,5000,7500)`, which equals 9773.38.

Subtracting the cost of the investment yields 9773.38-10000 or -226.62, and the investment would not be considered acceptable, because the net present value of the investment is less than its cost.

See also

IRR, MIRR, FV, NPER, PMT, PV, RATE

ODD

Syntax

ODD(*number*)

Description

ODD rounds *number* up to the nearest odd number.

Example

`ODD(2.2)` equals 3.

`ODD(-3.2)` equals -5.

`ODD(3)` equals 3.

See also

EVEN

OR

Syntax

OR(*logical1*,*logical2*,...*logical14*)

Description

OR returns TRUE if any of its arguments evaluate TRUE. If none of its arguments evaluate TRUE, OR returns FALSE.

OR accepts 1-14 arguments, which should be logical values, or references or arrays containing logical values. OR ignores blank cells and numbers and text within references and arrays. However, for the purposes of OR, non-zero numbers given as arguments are considered equivalent to TRUE, and zero given as an argument is considered equivalent to FALSE. A reference argument may be a range as well as a single cell. If no logicals are supplied, OR returns the #VALUE! error value.

Example

`OR(A1<1,A1>6)` equals TRUE if A1 contains a number outside the range (1,6).

`IF(OR(A1<1,A1>6),"Number is out of range","")` equals the empty text, "", if A1 is in the range (1,6), and "Number is out of range" if A1 contains a number outside this range.

`OR(A1:A3)` equals TRUE if A1:A3 contains the values FALSE, FALSE, and TRUE.

See also

AND, NOT, IF

PEARSON

Syntax

PEARSON(*y_array,x_array*)

Description

PEARSON computes the Pearson product moment correlation coefficient for the linear regression $y = a+bx$.

Y_array and *x_array* should be arrays or references with identical dimensions; *y_array* contains the dependent values, and *x_array* contains the independent values. PEARSON ignores blank cells and text and logical values within references and arrays. A number occurring in one array must be matched by a number at the same position in the complementary array; otherwise, PEARSON returns the #VALUE! error value. PEARSON returns the #INV! error value if fewer than 2 (x,y) data pairs are supplied.

The formula for PEARSON is:

$$r = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{\left(n\sum x^2 - (\sum x)^2\right)\left(n\sum y^2 - (\sum y)^2\right)}}$$

Example

`PEARSON({1.2,2.5,3.3,4.6},{1,2,3,4})` equals 0.9959.

See also

INTERCEPT, LINEST, RSQ, SLOPE, STEYX, TREND

PERMUT

Syntax

PERMUT(*number_of_items,number_chosen*)

Description

PERMUT returns the number of permutations of size *number_chosen* taken from a set containing *number_of_items* members. A permutation is an ordered sequence of items. If n = *number_of_items* and k = *number_chosen*, then the formula for PERMUT is:

$$P_{k,n} = \frac{n!}{(n-k)!}$$

Example

`PERMUT(5,3)` equals 60.

`PERMUT(5,5)` equals 120.

`PERMUT(10,4)` equals 5040.

See also

COMBIN

PI

Syntax

PI()

Description

PI returns the numeric constant *pi*, accurate to 15 decimal places.

PI takes no arguments; however, the parentheses *are* required.

Example

`PI()` equals 3.141592653589793.

See also

None.

PMT

Syntax

PMT(*rate,nper,pv,fv,type*)

Description

PMT returns the periodic payment component of an annuity.

Rate is the periodic interest rate.

Nper is the number of periods for the annuity.

Pv is the present value of the annuity, the amount it is worth today.

Fv is the future value of the annuity, the amount it will be worth after the last period.

Type indicates whether payments occur at the end of periods (ordinary annuity) or at the beginning (annuity due). Set *type* to 0 to designate an ordinary annuity and 1 to indicate an annuity due. If you omit *type*, it is considered equal to 0.

For a more complete description of the arguments, see PV.

Examples

Suppose you take out a four year car loan for £15,000. The annual interest rate is 11%, and interest is compounded monthly. What will your monthly payment be? The formula to compute this is:

`PMT(11%/12,48,15000)`, which equals -387.68.

The computed payment is negative because cash outflows are considered negative in the time value of money functions. Thus, your monthly payment is £387.68. In order to determine the amount of interest you will pay on the loan, add the loan amount to the product of the periodic payment and the number of periods. The total payment is -£387.68*48, or -£18,999.04, and you will pay -£18,999.04+£15,000 or -£3,999.04 as interest. By varying the interest rate and number of periods, you can easily see how these factors affect your monthly payment and the total interest you will pay.

Suppose instead you have £10,000 to deposit in an account today, and you wish to grow this to £100,000 at the end of 20 years. The account earns 8% annual interest, compounded monthly. How much do you have to deposit at the end of each month to achieve this goal? The formula to compute this amount is:

`PMT(8%/12,12*20,-10000,100000)`, which equals -86.13.

Thus, you will need to deposit £86.13 at the end of each month to accumulate £100,000 at the end of 20 years, based on an initial deposit of £10,000 in an account earning 8% annual interest, compounded monthly. Notice that *pv* is negative, because you are initially depositing money, which is a cash outflow. Also, this formula uses a *fv* argument, which is positive, because you will receive £100,000 at the end of the investment, a cash inflow.

See also

FV, NPER, PV, RATE, IPMT, PPMT

PPMT

Syntax

PPMT(*rate,per,nper,pv,* *fv,type*)

Description

PPMT returns the amount of the principal paid during a given period of an amortization schedule.

Rate is the periodic interest rate.

Per is the period for which you wish to determine the principal paid. It must range from 1 to *nper*.

Nper is the number of periods for the annuity.

Pv is the present value of the annuity, the amount it is worth today.

Fv is the future value of the annuity, the amount it will be worth after the last period.

Type indicates whether payments occur at the end of periods (ordinary annuity) or at the beginning (annuity due). Set *type* to 0 to designate an ordinary annuity and 1 to indicate an annuity due. If you omit *type*, it is considered equal to 0.

For a more complete description of the arguments, see PV.

Example

Suppose you take out a four year car loan for £15,000. The annual interest rate is 11%, and interest is compounded monthly. How much of the principal is paid off by the first payment? The formula to compute this is:

`PPMT(11%/12,1,48,15000)`, which equals -250.18.

Thus, the first payment pays off £250.18 of the principal.

See also

IPMT, FV, NPER, PMT, PV, RATE

PRODUCT

Syntax

PRODUCT(*number1*,*number2*,...*number14*)

Description

PRODUCT multiplies its arguments, returning the product.

PRODUCT accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. PRODUCT ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, PRODUCT returns zero.

Example

`PRODUCT(2,3,4)` equals 24.

`PRODUCT(A1:A4)` equals 24 if A1:A3 contains the values 2, 3, and 4, and A4 is blank.

See also

SUM, AVERAGE

PROPER

Syntax

PROPER(*text*)

Description

PROPER capitalises the first character in *text* and any character following a character that is not a letter.

Example

`PROPER("john doe")` equals "John Doe".

`PROPER("3 quarks for Muster Mark")` equals "3 Quarks For Muster Mark".

See also

CLEAN, TRIM

PV

Syntax

PV(*rate,nper,*_pmt,fv,type_)

Description

PV returns the present value of an investment. PV can compute the present value of an ordinary annuity or an annuity due, as well as performing lump-sum compound interest calculations.

Rate is the periodic interest rate. For example, a rate of 10% should be entered as 0.1 (Eureka, does, however, accept the "10%" notation, but remember this equates to 0.1). Since this is the *periodic* interest rate, you must ensure that the rate is consistent with the compounding frequency. For example, if interest is compounded monthly, and the annual rate is 10%, then the correct value for *rate* is 10%/12, or 0.83% per month.

Nper is the number of periods for the investment. For example, a 5-year annuity with monthly payments results in 5*12 or 60 periods.

Pmt is the periodic payment for an annuity. For example, if your monthly car payment is £200.00, then the correct *pmt* is -200. *Pmt* is negative here because this is a cash outflow. On the other hand, if someone is to pay you £200.00 each period, then the correct *pmt* is 200. *Pmt* is positive here because this is a cash inflow. If you omit *pmt*, it is considered zero, implying a lump sum calculation.

Fv is the future value of the investment. *Fv* should be positive if you are to receive money at the end of the investment. For example, if you are depositing money into a savings account, *fv* would be positive, since you would be able to withdraw money at the end of the investment. On the other hand, if you are paying money to satisfy a loan, *fv* would be negative from your point of view.

Type indicates whether payments occur at the end of periods (ordinary annuity) or at the beginning (annuity due). Set *type* to 0 to designate an ordinary annuity and 1 to indicate an annuity due. If you omit *type*, it is considered equal to 0.

At least one of *pmt* and *fv* must be specified. If you omit *pmt*, or set it equal to zero, PV performs a compound interest calculation for a lump sum. If you omit *fv*, it is considered equal to zero.

PV belongs to the "time value of money" family of functions, which includes FV, PMT, NPER, and RATE. When rate is not equal to zero, these functions are all computed by solving the following equation in terms of the other variables:

$$pv * (1 + rate)^{nper} + pmt * (1 + rate * type) * \left(\frac{(1 + rate)^{nper} - 1}{rate}\right) + fv = 0$$

This equation is based on the following assumptions:

- Each payment is for the same amount.
- The payments occur at regular intervals.
- The payment and compounding periods coincide.
- Cash inflows are represented by positive values, while cash outflows are represented by negative values.

If *rate* is zero or omitted, the equation used is:

$$(pmt * nper) + pv + fv = 0$$

Examples

Suppose you have the opportunity to receive £100 at the end of each month for the next two years. This investment will cost you £2,000, and you do have this much money on hand. To determine whether this is a good investment, you want to compute the present value—the amount you would have to deposit today in an account paying identical interest, such that you could withdraw £100 at the end of each month over the next two years. Thus, the formula is:

`PV(10%/12,24,100)`, which equals -2167.09.

The periodic interest rate is 10%/12 or 0.83%, the number of periods is 24 (2 years * 12 months per year), the payment is 100 (positive, since you are receiving money), and the future value is omitted and thus zero, because the funds should be exhausted at the end of two years. The present value is negative, because you would have to pay this amount today, and cash outflows are negative. Thus, you see that you would have to deposit £2,167.09 today in order to withdraw £100 at the end of each month over the next two years from an account paying 10% annual interest, compounded monthly, but the investment opportunity pays you the same amount at the end of each month and only requires you to pay £2,000 today. By this criteria, you can conclude this is an acceptable investment.

Suppose instead you want to deposit some money today in a savings account earning 10% annual interest, compounded monthly, and after two years, with no interim deposits or withdrawals, you wish to withdraw £3,000. How much money must you deposit today to accomplish this goal? Unlike the example above, this is not an annuity calculation; rather, it is a compound interest calculation on a lump sum. Thus, the formula is:

`PV(10%/12,24,,3000)`, which equals -2458.23.

The *rate* and *nper* arguments are as described under the first example. However, notice this formula omits the *pmt* argument; PV considers it zero, which is appropriate since you are not making periodic payments. Since you wish to withdraw £3,000 at the end of 24 months, you set the future value argument, *fv*, to 3000; *fv* is positive because you will be receiving money. Finally, the present value is again negative, because you will have to pay £2,458.23 today in order to be able to withdraw £3,000 at the end of two years at the given interest rate and compounding frequency.

See also

FV, NPER, PMT, RATE, IPMT, PPMT

QUARTILE

Syntax

QUARTILE(*array,quartile*)

Description

QUARTILE returns the value of the indicated *quartile* of the data set *array*. QUARTILE's result depends on the value of *quartile* as follows:

If *quartile* is	QUARTILE returns
0	The smallest value (MIN).
1	The first quartile.
2	The second quartile (MEDIAN).
3	The third quartile.
4	The largest value (MAX).

The function names in parentheses are equivalent to QUARTILE for the indicated *quartile* values and should be preferred.

Example

`QUARTILE({1,2,3,4,5,6,7},1)` equals 2.

See also

IQRANGE, LARGE, MAX, MEDIAN, MIN, SMALL

RADIANS

Syntax

RADIANS(*angle_in_degrees*)

Description

RADIANS converts *angle_in_degrees* to radians.

Example

`RADIANS(180)` equals *pi*.

See also

DEGREES

RAND

Syntax

RAND()

Description

RAND returns a uniformly distributed, pseudo-random number n in the range $0 \leq n < 1$. RAND returns a different value each time it is called.

RAND is a *volatile* function; that is, formulas using RAND are recalculated upon data entry and are included in all other recalculations. However, by itself, RAND will *not* cause a formula to be recalculated upon loading a file using it.

RAND takes no arguments; however, the parentheses *are* required.

Example

`RAND()` (might) equal 0.152134.

See also

RANDBETWEEN, RANDNORMAL

RANDBETWEEN

Syntax

RANDBETWEEN(*low_limit,high_limit*)

Description

RANDBETWEEN returns a uniformly distributed, pseudo-random integer n in the range $low\_limit \leq n \leq high\_limit$. RANDBETWEEN returns a different value each time it is called.

RANDBETWEEN is a *volatile* function; that is, formulas using RANDBETWEEN are recalculated upon data entry and are included in all other recalculations. However, by itself, RANDBETWEEN will *not* cause a formula to be recalculated upon loading a file using it.

Example

`RANDBETWEEN(1, 6)` might equal 3 on a given trial, but it will return 1, 2, 3, 4, 5, and 6 in approximately equal proportions over a large number of recalculations.

See also

RAND, RANDNORMAL

RANDNORMAL

Syntax

RANDNORMAL(*mean,sigma*)

Description

RANDNORMAL returns a pseudo-random number following a normal (Gaussian) distribution with mean *mean* and standard deviation *sigma*. RANDNORMAL returns a different value each time it is called.

Sigma should be a positive number.

RANDNORMAL is a *volatile* function; that is, formulas using RANDNORMAL are recalculated upon data entry and are included in all other recalculations. However, by itself, RANDNORMAL will *not* cause a formula to be recalculated upon loading a file using it.

Example

`RANDNORMAL(0,1)` might equal 0.11 on a given trial, but the average of the random numbers over a large number of calculations will approach 0, and the standard deviation will approach 1.

See also

RAND, RANDBETWEEN

RANGE

Syntax

RANGE(*number1*,*number2*,...*number14*)

Description

RANGE returns the difference between the largest and smallest values in its arguments. Thus, RANGE is equivalent to MAX(*args*)-MIN(*args*).

RANGE accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. RANGE ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If fewer than two values are supplied, RANGE returns the #INV! error value.

Example

`RANGE({1,2,3,4})` equals 3.

`RANGE(A1:A2,3,4)` equals 3 if A1:A2 contains 1 and 2.

See also

MAX, MIN, IQRANGE

RATE

Syntax

RATE(*nper*,*pmt*,***pv***,*fv*,*type*,*guess*)

Description

RATE returns the periodic interest rate for an annuity or compound interest on a lump sum.

Nper is the number of periods.

Pmt is the periodic payment for an annuity.

Pv is the present value of the investment.

Fv is the future value of the investment.

Type indicates whether payments occur at the end of periods (ordinary annuity) or at the beginning (annuity due). Set *type* to 0 to designate an ordinary annuity and 1 to indicate an annuity due. If you omit *type*, it is considered equal to 0.

Guess is your guess as to what the periodic rate actually is. If you omit *guess*, it is considered equal to 0.1.

For a more complete description of the arguments, see PV.

If *pmt* is not zero, the calculation is for an annuity, and RATE uses an iterative technique to find the interest rate. RATE terminates successfully if the absolute value of the difference between its successive approximations to the true rate becomes less than 10^{-7}. RATE performs at most 20 iterations, returning the #NUM! error value if this limit is exceeded. If this should happen, try adjusting the *guess* argument. RATE usually will converge to a solution with a *guess* value between 0 and 1.

Examples

Suppose you take out a loan for £10,000, which you want to pay off over 5 years, making equal monthly payments. If interest on the loan will also be compounded monthly, what will the interest rate be if you are to pay £200 per month? The formula to compute this is:

`RATE(5*12,-200,10000)`, which equals 0.006183.

Thus the periodic interest rate is 0.006183, or 0.6183%. Notice that this is the *periodic* rate, and since one period is equivalent to one month, you must multiply this rate by 12 to determine the annual rate, which is 0.6183%*12 or 7.42%. The *pmt* argument is negative, because this is a cash outflow, and outflows are always negative. The *pv* argument is positive, because you initially receive this amount, and cash inflows are always positive.

Suppose instead you have £10,000 to deposit in an account today, and you want it to double to £20,000 over 8 years, with no interim deposits or withdrawals. This is not an annuity calculation but rather a compound interest calculation on a lump sum. If interest is to be compounded monthly, what interest rate does this require? The formula to compute the periodic rate is:

`RATE(8*12,,-10000,20000)`, which equals 0.007246.

Thus, a periodic rate of 0.007246 or 0.7246% will allow you to double the initial deposit in 8 years. Again, this is a periodic rate, and since one period is equal to one month, the annual rate is 0.7246%*12 or 8.7%. As long as you can find an account earning at least this much annual interest, compounded monthly, you can achieve your goal.

See also

FV, NPER, PMT, PV, RATE

REPLACE

Syntax

REPLACE(*text,start_position,num_chars,replace_text*)

Description

REPLACE replaces a portion of *text* with *replace_text* and returns the modified *text*.

Text is the text string upon which REPLACE will act.

Start_position is the position within *text* where the replace operation will begin. It should be a number between 1 and the length of *text*.

Num_chars is the number of characters in *text* beginning at *start_position* that will be replaced. In effect, *num_chars* characters in *text* are deleted.

Replace_text is the text which will replace the deleted part of *text*.

Example

`REPLACE("Yearly",1,4,"Month")` equals "Monthly".

`REPLACE("Password: 123456",11,6,"***")` equals "Password: ***".

See also

FIND, SEARCH, SUBSTITUTE, TRIM

REPT

Syntax

REPT(*text,num_reps*)

Description

REPT repeats *text num_reps* times and returns the resulting text. If *num_reps* is zero, REPT returns the empty text, "". If the return value exceeds 255 characters in length, it is truncated.

Example

`REPT("10",2)` equals "1010".

`"Wow"&REPT("!",Surprise_degree)` equals "Wow!!!!" if Surprise_degree is a name equal to 4.

See also

None.

REVERSE

Syntax

REVERSE(*text_or_array*)

Description

If *text_or_array* is text, REVERSE reverses the order of the characters in text. When passed an array, REVERSE reverses the order of the elements in the array. The array must be a vector, that is an array with only one row or only one column; otherwise, REVERSE returns the #INV! error value.

Example

`REVERSE("abc")` equals "cba".

`IF(REVERSE(A1)=A1,"Is a palindrome.")` equals "Is a palindrome" if A1 contains text that is spelled the same backwards as forwards.

`REVERSE({1,2,3})` equals {3,2,1}.

`REVERSE(LINEST(Y_vals,X_vals))` equals the vector of coefficients for the linear regression in reverse order, that is $\{b, x_1, x_2 \ldots x_n\}$.

See also

None.

RIGHT

Syntax

RIGHT(*text*,*number_of_characters*)

Description

RIGHT returns the right-most *number_of_characters* from *text*.

If *number_of_characters* is omitted, RIGHT returns the last character in *text*. If *number_of_characters* exceeds the length of *text*, RIGHT returns *text*.

Example

`RIGHT("Part #57-A32",6)` equals "57-A32".

See also

LEFT

ROUND

Syntax

ROUND(*number*,_decimal_places_**)**

Description

ROUND rounds *number* to the number of decimal places indicated by *decimal_places*.

If *decimal_places* is omitted, it is treated as zero.

Decimal_places may be negative, in which case rounding occurs to the left of the decimal point.

Example

`ROUND(3.5)` equals 4.

`ROUND(-3.5)` equals -4.

`ROUND(15.746,2)` equals 15.75.

`ROUND(157,-1)` equals 160.

`ROUND(157,-2)` equals 200.

`ROUND(157,-3)` equals 0.

See also

TRUNC, INT

ROW

Syntax

ROW(*reference*)

Description

ROW returns the row number of *reference*. If *reference* is omitted, ROW returns the row number of the cell containing the formula. If *reference* is for a range, and the formula is entered as an array formula, ROW returns the row numbers of *reference* as a vertical array.

ROW returns the #INV! error value if *reference* is a multiple area or not a reference.

Example

`ROW()` equals 1 if used in a formula in row 1 of a worksheet.

`ROW(C2)` equals 2.

`{ROW(A1:C5)}` equals {1;2;3;4;5}.

See also

COLUMN, COLUMNS, ROWS

ROWS

Syntax

ROWS(*array*)

Description

ROWS returns the number of rows in an array or reference.

ROWS returns the #INV! error value if *array* is a multiple area or not an array or reference.

Example

`ROWS(A1:B5)` equals 5.

`ROWS({1,2,3;4,5,6})` equals 2.

See also

COLUMN, COLUMNS, ROW

RSQ

Syntax

RSQ(*y_array,x_array*)

Description

RSQ returns the r^2 statistic for the linear regression $y = a+bx$. RSQ is equivalent to PEARSON(Y,X)^2.

Y_array and *x_array* should be arrays or references with identical dimensions; *y_array* contains the dependent values, and *x_array* contains the independent values. RSQ ignores blank cells and text and logical values within references and arrays. A number occurring in one array must be matched by a number at the same position in the complementary array; otherwise, RSQ returns the #VALUE! error value. RSQ returns the #INV! error value if fewer than 2 (x,y) data pairs are supplied.

Example

`RSQ({1.2,2.5,3.3,4.6},{1,2,3,4})` equals 0.9918.

See also

INTERCEPT, LINEST, PEARSON, SLOPE, STEYX, TREND

SEARCH

Syntax

SEARCH(*search_text*,*within_text*,*start_at_number*)

Description

SEARCH performs a case-sensitive search for *search_text* within *within_text*, starting at character position *start_at_number*. SEARCH returns the character position of the first match. SEARCH supports the following wildcard characters:

*	Matches zero or more characters.
?	Matches exactly one character.
[]	Defines a *character class* and matches exactly one of the characters enclosed by the brackets. To match a range of characters, use the notation *first-last*, where *first* is the first character and *last* the last character you wish to match, in the order of the ASCII character set. Finally, if the first character in the class is the circumflex, '^', then only a character not occurring in the class will be matched. This is called a *complemented* character class.

To suppress the special meaning of a wildcard character, preface it with the tilde, '~'.

The first character in *text* is numbered 1, and if you omit *start_at_number*, SEARCH begins the search at the beginning of *within_text*. If there is no match, SEARCH returns the #VALUE! error value. SEARCH returns #INV! if *start_at_number* is less than one or greater than the length of *within_text*. If *find_text* is the empty text, "", SEARCH returns 1, unless *within_text* is also the empty text, in which case it returns #INV!.

Example

`SEARCH("b","abcdef")` equals 2.

`SEARCH("b*e","abcdef")` equals 2.

`SEARCH("b?e","abcdef")` equals #VALUE!.

`SEARCH("b[a-z0-9]d","ABCDEF")` equals 2.

See also

FIND

SECOND

Syntax

SECOND(*serial_number*)

Description

SECOND returns the second of *serial_number*, an integer falling in the range 0-59.

You may use a textual time rather than a number; Eureka converts the text to a serial number automatically. However, text in references and arrays is *not* converted and generates a #VALUE! error when used with SECOND.

Example

`SECOND("3:30:45 PM")` equals 45.

`SECOND(33000.34)` equals 36.

See also

DAY, MONTH, YEAR, HOUR, MINUTE

SIGN

Syntax

SIGN(*number*)

Description

SIGN returns a number indicating whether its argument is positive, negative, or zero. SIGN returns 1 if *number* is positive, -1 if *number* is negative, and 0 if *number* equals zero.

Example

`SIGN(4.5)` equals 1.

`SIGN(-2.4)` equals -1.

`SIGN(0)` equals 0.

See also

ABS

SIN

Syntax

SIN(*angle_in_radians*)

Description

SIN returns the sine of *angle_in_radians*.

Example

`SIN(0)` equals 0.

`SIN(PI()/2)` equals 1.

See also

ASIN, COS, TAN

SINH

Syntax

SINH(*number*)

Description

SINH returns the hyperbolic sine of *number*. SINH is defined by the formula:

$$\sinh x = \frac{e^x - e^{-x}}{2}$$

Example

`SINH(0)` equals 0.

`SINH(2)` equals 3.6269.

See also

ASINH

SLN

Syntax

SLN(*cost,salvage,life*)

Description

SLN returns the straight-line depreciation of an asset for a single period.

Cost is the initial cost of the asset.

Salvage is the salvage value, the amount the asset will be worth at the end of the depreciation.

Life is the number of periods over which the asset will be depreciated.

The formula for SLN is:

$$\text{SLN} = \frac{(cost - salvage)}{life}$$

Example

Suppose you have purchased a piece of equipment for £10,000, and you will be able to depreciate it for 5 years. If the salvage value is £4,000, then the straight-line depreciation allowance for each of the next 5 years is:

`SLN(10000,4000,5)` which equals 1200.

See also

DB, DDB, SYD, VDB

SLOPE

Syntax

SLOPE(*y_array,x_array*)

Description

SLOPE computes the value *b* in the linear regression $y = a+bx$.

Y_array and *x_array* should be arrays or references with identical dimensions. SLOPE ignores blank cells and text and logical values within references and arrays. A number occurring in one array must be matched by a number at the same position in the complementary array; otherwise, SLOPE returns the #VALUE! error value. SLOPE returns the #INV! error value if both *y_array* and *x_array* are empty.

The formula for SLOPE is:

$$b=\frac{n\sum xy-\left(\sum x\right)\left(\sum y\right)}{n\sum x^2-\left(\sum x\right)^2}$$

Example

`SLOPE({1.2,2.5,3.3,4.6},{1,2,3,4})` equals 1.1.

See also

INTERCEPT, LINEST, TREND

SMALL

Syntax

SMALL(*array,k*)

Description

SMALL returns the k^{th} smallest value occurring in *array*.

Array should be an array or reference containing numbers. SMALL ignores blank cells and text and logical values within arrays and references.

K should be a value ranging from 1 to the number of values in *array*. To obtain the smallest value, set *k* to 1, to obtain the second smallest value, set *k* to 2, and so on.

If *array* is empty, or *k* is < 1 or exceeds the number of values in *array*, SMALL returns the #INV! error value.

Example

`SMALL({2,1,5,4},2)` equals 2.

See also

LARGE, MAX, MEDIAN, MIN, QUARTILE

SQRT

Syntax

SQRT(*number*)

Description

SQRT returns the positive square root of *number*, which must be a non-negative number. SQRT returns #INV! if *number* is negative.

Example

`SQRT(2)` equals 1.4142, to 4 decimal places.

See also

None.

STDEV

Syntax

STDEV(*number1*,*number2,...number14*)

Description

STDEV computes the sample standard deviation, which is the positive square root of the sample variance. The sample standard deviation is defined by the formula:

$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}$$

STDEV accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. STDEV ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. At least two values are required to compute STDEV. If fewer than two values are supplied, STDEV returns the #DIV/0! error value.

Example

`STDEV(2.4,3.2,2.8)` equals 0.4000.

`STDEV(A1:A3)` equals 0.4000 if A1:A3 contains the values 2.4, 3.2, and 2.8.

See also

STDEVP, VAR, VARP

STDEVP

Syntax

STDEVP(*number1* *,number2,...number14*)

Description

STDEVP computes the population standard deviation, which is the positive square root of the population variance. The population standard deviation is defined by the formula:

$$\sigma = \sqrt{\frac{\sum (x-\bar{x})^2}{n}}$$

STDEVP accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. STDEVP ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. At least two values are required to compute STDEVP. If fewer than two values are supplied, STDEVP returns the #DIV/0! error value.

Example

`STDEVP(2.4,3.2,2.8)` equals 0.3266.

`STDEVP(A1:A3)` equals 0.3266 if A1:A3 contains the values 2.4, 3.2, and 2.8.

See also

STDEV, VAR, VARP

STEYX

Syntax

STEYX(*y_array,x_array*)

Description

STEYX computes the standard error of the linear regression $y = a+bx$.

Y_array and *x_array* should be arrays or references with identical dimensions; *y_array* contains the dependent values, and *x_array* contains the independent values. STEYX ignores blank cells and text and logical values within references and arrays. A number occurring in one array must be matched by a number at the same position in the complementary array; otherwise, STEYX returns the #VALUE! error value. STEYX returns the #INV! error value if fewer than 3 (x,y) data pairs are supplied.

The formula for STEYX is:

$$S_{y \cdot x} = \sqrt{\frac{1}{n(n-2)}\left(n\sum y^2 - \left(\sum y\right)^2 - \frac{\left(n\sum xy - \left(\sum x\right)\left(\sum y\right)\right)^2}{n\sum x^2 - \left(\sum x\right)^2}\right)}$$

Example

`STEYX({1.2,2.5,3.3,4.6},{1,2,3,4})` equals 0.1581.

See also

INTERCEPT, LINEST, PEARSON, RSQ, SLOPE, TREND

SUBSTITUTE

Syntax

SUBSTITUTE(*text,match_text,replace_text,*instance_number**)**

Description

SUBSTITUTE replaces part of *text* that matches *match_text* with *replace_text* and returns the modified *text*.

Text is the text string upon which SUBSTITUTE will act.

Match_text is the text which will be replaced if it occurs in *text*. The comparison is case-sensitive.

Replace_text is the text which will replace text agreeing with *match_text* in *text*.

Instance_number designates the particular instance of *match_text* that should be replaced. For example, if *match_text* occurs 3 times in *text*, you can set *instance_number* to 1, 2, or 3, to replace the first, second, or third instance, respectively. If you omit *instance_number*, SUBSTITUTE replaces all instances.

Example

`SUBSTITUTE("The weather is rainy.","rainy",`
`"sunny")` equals "The weather is sunny."

`SUBSTITUTE("1st Quarter","1st","2nd",1)` equals "2nd Quarter."

`SUBSTITUTE(PATH,":",",")` replaces all instances of ':' in PATH with ','.

See also

FIND, REPLACE, SEARCH, TRIM

SUM

Syntax

SUM(*number1*,*number2,...number14*)

Description

SUM adds its arguments and returns the total.

SUM accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. SUM ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, SUM returns zero.

Example

`SUM(2.4,3.2,2.8)` equals 8.4.

`SUM(A1:A3)` equals 8.4 if A1:A3 contains the values 2.4, 3.2, and 2.8.

See also

AVERAGE, PRODUCT

SUMSQ

Syntax

SUMSQ(*number1*,*number2,...number14*)

Description

SUMSQ returns the sum of the squares of its arguments.

SUMSQ accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. SUMSQ ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. If no values are supplied, SUMSQ returns zero.

Example

`SUMSQ(2,4)` equals 20.

See also

SUM

SUMX2MY2

Syntax

SUMX2MY2(*x_array,y_array*)

Description

SUMX2MY2 returns the sum of the differences of the squares of corresponding elements in the two arrays. The formula used is:

$$\sum x^2 - y^2$$

X_array and *y_array* should contain the same number of elements, and they should also be the same shape.

Example

`SUMX2MY2({1,2,3},{2,3,4})` equals -15.

See also

SUMX2PY2, SUMXMY2

SUMX2PY2

Syntax

SUMX2PY2(*x_array,y_array*)

Description

SUMX2PY2 returns the sum of the squares of corresponding elements in the two arrays. The formula used is:

$$\sum x^2 + y^2$$

X_array and *y_array* should contain the same number of elements, and they should also be the same shape.

Example

`SUMX2PY2({1,2,3},{2,3,4})` equals 43.

See also

SUMX2MY2, SUMXMY2

SUMXMY2

Syntax

SUMXMY2(*x_array,y_array*)

Description

SUMXMY2 returns the sum of the squared differences of corresponding elements in the two arrays. The formula used is:

$$\sum (x - y)^2$$

X_array and *y_array* should contain the same number of elements, and they should also be the same shape.

Example

`SUMXMY2({1,2,3},{2,3,4})` equals 3.

See also

SUMX2MY2, SUMX2PY2

SYD

Syntax

SYD(*cost,salvage,life,per*)

Description

SYD returns the sum-of-the-years' digits depreciation of an asset for a specified period.

Cost is the initial cost of the asset.

Salvage is the salvage value, the amount the asset will be worth at the end of the depreciation.

Life is the number of periods over which the asset will be depreciated.

Per is the period for which you want the depreciation.

The formula for SYD is:

$$\mathrm{SYD} = \frac{(cost - salvage) * (life - period + 1)}{((life * (life + 1))/2)}$$

Example

Suppose you have purchased a piece of equipment for £10,000, and you will be able to depreciate it for 5 years. If the salvage value is £4,000, then the SYD depreciation for the first year is:

`SYD(10000,4000,5,1)` which equals 2000.

The depreciation for the second year would be:

`SYD(10000,4000,5,2)` which equals 1600.

See also

DB, DDB, SLN, VDB

T

Syntax

T(*value*)

Description

T returns the text *value* refers to.

If *value* is or refers to text, T returns that text. Otherwise, T returns the empty text, "".

Example

`T(A1)` equals "some text" if A1 contains the text "some text".

`T(1.5)` equals "".

See also

N

TAN

Syntax

TAN(*angle_in_radians*)

Description

TAN returns the tangent of *angle_in_radians*.

Example

`TAN(PI()/4)` equals 1.

See also

ATAN, ATAN2, SIN, COS

TANH

Syntax

TANH(*number*)

Description

TANH returns the hyperbolic tangent of *number*. TANH is defined by the formula:

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Example

`TANH(0)` equals 0.

`TANH(1)` equals 0.7616.

See also

ATANH

TEXT

Syntax

TEXT(*value,format_text*)

Description

TEXT formats *value* according to *format_text* and returns the resulting text. *Format_text* uses the formatting codes described under *Number Formatting* in the reference section; however, certain format codes are not available to TEXT. These include the * symbol, for repeating a character to fill a column, the underscore symbol, for inter-character spacing, and colour options. These codes are ignored when used in the *format_text* argument.

Example

`TEXT(A1,"d-mmm-yy")` equals "22-Sep-92", if A1 contains the serial number 33869.

`TEXT(15.2,"0.0000")` equals "15.2000".

`"The number is"&TEXT(A1,"0.0000")&"."` equals "The number is 15.2000." if A1 contains 15.2.

See also

FIXED, DOLLAR

TIME

Syntax

TIME(*hour,minute,second*)

Description

TIME returns the serial number of the indicated time.

Hour should range from 0-23.

Minute should range from 0-59.

Second should range from 0-59.

TIME returns #INV! if any of its arguments are less than 0; however, if any arguments exceed the ranges given above, TIME *will* function properly, in effect "rolling over" into successive minutes and hours. TIME will always return a serial number less than 1; any integer (date) component that might be generated due to large arguments is discarded.

Example

`TIME(15,0,0)` equals 0.625, which corresponds to 3:00:00 PM.

See also

DATE

TIMEVALUE

Syntax

TIMEVALUE(*time_text*)

Description

TIMEVALUE returns the serial number corresponding to *time_text*, which should be a textual time in one of Eureka's built-in time formats. These formats are also recognised when textual time are used directly in formulas and when entered into cells.

Example

`TIMEVALUE("6:00 PM")` equals 0.75.

`TIMEVALUE(A1)` equals 0.75 if A1 contains the text "6:00 PM".

`"6:00 PM"+1/8` equals 0.875 (9:00 PM).

`A1+1/8` equals #VALUE! if A1 contains the text "6:00 PM", but `TIMEVALUE(A1)+1` equals 0.875.

See also

DATEVALUE, VALUE

TODAY

Syntax

TODAY()

Description

TODAY returns the serial number representing today's date, according to your computer's clock.

Unlike NOW, TODAY returns an integer value, that is, a serial number with no time component. TODAY is a *volatile* function; that is, formulas using TODAY are recalculated upon data entry and are included in all other recalculations. In addition, when you load a file, Eureka recalculates all formulas using TODAY.

TODAY takes no arguments; however, the parentheses *are* required.

Example

`TODAY()` equals 33871, if today's date is 24-Sep-92.

See also

NOW

TRANSPOSE

Syntax

TRANSPOSE(*array*)

Description

TRANSPOSE returns the transpose of *array*. The transpose of a matrix **A**, $\mathbf{A}^t$, is given by the formula:

$$d_{ij} = a_{ji}$$

Put simply, the rows and columns are swapped, and the resulting array has as many columns as the original has rows and as many rows as the original has columns.

Example

`TRANSPOSE({1,2,3;4,5,6})` equals {1,4;2,5;3,6}.

See also

MINVERSE, MMULT

TREND

Syntax

TREND(*y_array*,x_array,*new_x_values*,*const*)

Description

TREND returns the y-estimates for the values in *new_x_values* by first computing the multiple regression $y=b+m_1x_1+m_2x_2+\ldots+m_nx_n$, or in its simplest form, the linear regression $y=b+mx$.

Y_array should be an array or cell range containing the dependent values.

X_array should be an array or cell range containing the independent values. If you omit *x_array*, TREND considers it to be valued as {1, 2, 3, ..., *n*}, but in the same size and shape as *y_array*.

New_x_values can be a value, reference, or array. If you omit *new_x_values*, TREND considers it to be valued as {1, 2, 3, ..., *n*}, but in the same size and shape as *y_array*, unless *x_array* was specified, in which case TREND uses a copy of *x_array* for *new_x_values*. If *new_x_values* is an array, TREND returns an array of the same size and shape containing the estimates.

Const is a logical argument which controls whether TREND includes the constant term *b* in its calculations. If *const* is TRUE or omitted, then the constant term is included. If *const* is FALSE or 0, then TREND excludes the constant term from its calculations (this is the *zero* or *no-intercept model*).

For a more detailed description of TREND's arguments, refer to LINEST.

Example

Suppose yearly profits for a company have been 180,000, 200,000, 223,000, 238,000, and 265,000 over the last 5 years. If the company is confident this trend will continue into the near future, it could predict profits for the following year as follows:

```
TREND({180000,200000,223000,238000,265000},,6)
```

which equals 283,600. Thus, the company can anticipate profits around 283,600 in the next year, *if* its assumption that the trend will continue is valid.

See also

GROWTH, INTERCEPT, LINEST, LOGEST, PEARSON, RSQ, SLOPE, STEYX

TRIM

Syntax

TRIM(*text*)

Description

TRIM removes all leading and trailing spaces from *text* and folds multiple spaces between words into single spaces.

Example

`TRIM("  1st   Quarter   Summary  ")` equals "1st Quarter Summary".

See also

CLEAN, PROPER

TRUE

Syntax

TRUE()

Description

The TRUE function returns the logical value TRUE.

TRUE takes no arguments; however, the parentheses *are* required. Note that you can also use the logical values TRUE and FALSE directly in formulas.

See also

FALSE

TRUNC

Syntax

TRUNC(*number*,*decimal_places*)

Description

TRUNC truncates *number* to the number of decimal places indicated by *decimal_places*.

If *decimal_places* is omitted, it is treated as zero.

Decimal_places may be negative, in which case truncating occurs to the left of the decimal point.

Example

`TRUNC(3.5)` equals 3.

`TRUNC(-3.5)` equals -3.

`TRUC(15.746,2)` equals 15.74.

`TRUNC(157,-1)` equals 150.

`TRUNC(157,-2)` equals 100.

`TRUNC(157,-3)` equals 0.

See also

ROUND, INT

TYPE

Syntax

TYPE(*value*)

Description

TYPE returns a numeric code representing the data type of *value*, according to the following table.

If *value* is	TYPE returns
Numeric	1
Text	2
Logical	4
Error	16
Array	64

TYPE returns #INV! if *value* is a reference to an empty cell.

Example

`TYPE(1.57)` equals 1.

`TYPE({1,"a";TRUE,#NUM!})` equals 64.

See also

ERROR.TYPE

UPPER

Syntax

UPPER(*text*)

Description

UPPER converts all lower-case characters in *text* to upper-case and returns the modified text.

Example

`UPPER("a.b.C.D.")` equals "A.B.C.D."

See also

LOWER

VALUE

Syntax

VALUE(*text*)

Description

VALUE returns the number corresponding to *text*, which should be a textual value in one of Eureka's built-in numeric formats. These formats are also recognised when textual values are used directly in formulas and when entered into cells.

Example

`VALUE("£1,000.25")` equals 1000.25.

`VALUE("22-Sep-92")` equals 33869.

`VALUE(A1)` equals 1000.25 if A1 contains the text "£1,000.25".

`"£1,000.25"+100` equals 1100.25.

`A1+100` equals #VALUE! if A1 contains the text "£1,000.25", but `VALUE(A1)+100` equals 1100.25.

See also

DATEVALUE, TIMEVALUE

VAR

Syntax

VAR(*number1*,*number2,...number14*)

Description

VAR computes the sample variance, which is a measure of dispersion about the mean. The sample variance is defined by the formula:

$$s^2 = \frac{\sum (x - \bar{x})^2}{n - 1}$$

VAR accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. VAR ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. At least two values are required to compute VAR. If fewer than two values are supplied, VAR returns the #DIV/0! error value.

Example

`VAR(2.4,3.2,2.8)` equals 0.1600.

`VAR(A1:A3)` equals 0.1600 if A1:A3 contains the values 2.4, 3.2, and 2.8.

See also

VARP, STDEV, STDEVP

VARP

Syntax

VARP(*number1*,*number2,...number14*)

Description

VARP computes the population variance, which is a measure of dispersion about the mean. The population variance is defined by the formula:

$$\sigma^2 = \frac{\sum (x - \bar{x})^2}{n}$$

VARP accepts 1-14 arguments, which should be numbers, or references or arrays containing numbers. VARP ignores blank cells and text and logical values within references and arrays. A reference argument may be a range as well as a single cell. At least two values are required to compute VARP. If fewer than two values are supplied, VARP returns the #DIV/0! error value.

Example

`VARP(2.4,3.2,2.8)` equals 0.1067.

`VARP(A1:A3)` equals 0.1067 if A1:A3 contains the values 2.4, 3.2, and 2.8.

See also

VAR, STDEV, STDEVP

VDB

Syntax

VDB(*cost,salvage,life,start_period,end_period,*factor,no_switch**)**

Description

VDB returns the depreciation of an asset for the indicated period using the double-declining balance method. VDB stands for "variable declining balance."

Cost is the inital cost of the asset.

Salvage is the salvage value, the amount the asset will be worth after all depreciation has been taken. VDB adjusts the depreciation for the final period as necessary so as to avoid depreciating an asset below its salvage value.

Life is the number of periods over which the asset will be depreciated.

Start_period is the starting period for which you wish the depreciation.

End_period is the ending period for which you wish the depreciation.

Factor controls the rate at which the balance declines. If you omit *factor*, VDB considers it equal to 2, resulting in a double-declining balance calculation.

No_switch controls whether VDB switches to straight-line depreciation at the point at which this exceeds the declining balance depreciation. If *no_switch* is zero or omitted, VDB does switch to straight-line depreciation, should it exceed the declining balance depreciation. If *no_switch* is non-zero, then VDB does not make this switch.

Example

Suppose you have purchased a piece of equipment for £10,000, and it is to be depreciated over 10 years. If the salvage value is £1,000, then the first year's depreciation is:

`VDB(10000,1000,10,0,1)` which equals 2000.

The first months' depreciation is:

`VDB(10000,1000,120,0,1)` which equals 166.67.

The total depreciation is:

`VDB(10000,1000,120,0,120)` which equals 9000.

If instead you had purchased the asset in the middle of the first year, then the first six months' depreciation would be:

`VDB(10000,1000,10,0,0.5)` which equals 1000.

The last 9 1/2 years' depreciation would then be:

`VDB(10000,1000,10,0.5,10)` which equals 8000. Notice that the sum of this example and the preceding one is £9,000, the total allowed depreciation.

See also

DB, DDB, SLN, SYD

VLOOKUP

Syntax

VLOOKUP(*lookup_value,array,column_offset*)

Description

VLOOKUP performs a "vertical lookup" on *array*. VLOOKUP, searches for *lookup_value* in the first column of *array*, and upon finding a match, returns the value in *array* in the same row as the match but in the column given by *column_offset*.

Lookup_value is the value you wish to find. It should be text, numeric, or logical. Text comparisons are case-insensitive, and wildcard characters are matched literally (that is, they do not act as wildcards).

Array is the array or cell range which VLOOKUP will search. VLOOKUP searches for *lookup_value* in the first column of *array*, and VLOOKUP expects the values in the first column to be sorted in ascending order, according to Eureka's collating scheme, in which the order is: numbers, text, logicals, error values, and blank cells.

Column_offset is a 1-based offset from the first column in *array*. That is, if *array* is the range B1:C5, then a *column_offset* of 1 refers to column B. When VLOOKUP finds a match for *lookup_value* in the first column of *array*, it returns the value in the same row as the match, but in the column indicated by *column_offset*.

If VLOOKUP fails to find an exact match for *lookup_value*, it matches the largest value not greater than *lookup_value*. If no such value exists, VLOOKUP returns the #N/A error value. If *column_offset* is < 1, VLOOKUP returns #VALUE!, and if *column_offset* is greater than the number of columns in *array*, VLOOKUP returns the #REF! error value.

Examples

	A	B	C	D	E
1	**Floppy disk pricing**				
2		1-49	50-99	100-499	500+
3	Generic	0.24	0.22	0.19	0.17
4	Name brand	0.48	0.43	0.39	0.35
5	Premium	0.30	0.27	0.24	0.22

The worksheet above defines an imaginary company's discounted pricing scheme for floppy disks. If we name A3:E5 Table, then:

`VLOOKUP("Name brand",Table,3)` equals 0.43.

If cell A20 holds the disk type, and cell B20 holds the quantity ordered, then if A20 equals "Generic" and B20 equals 100,

`VLOOKUP(A20,Table,(B20>0)+(B20>49)+(B20>99)+(B20>499)+1)` equals `VLOOKUP(A20,Table,4)` equals 0.19. This formula takes advantage of the fact that the result of a logical operator is either TRUE or FALSE, which translate to 1 and 0, respectively, when used in expressions. A term `(B20>0)` is called a "logical switch," and in this example the first three logical switches are TRUE, while the last one is FALSE, so the last argument becomes TRUE+TRUE+TRUE+FALSE+1, which in turn becomes 1+1+1+0+1, which equals 4, the correct *column_offset* for a quantity between 100 and 499.

As a final remark, notice that we omitted row 2, which contains the unit ranges, from the range name Table. We had to omit this row, because A2 is blank, and including it would violate Eureka's collating order, described above, causing VLOOKUP to fail, as a blank cell would precede text cells in the first column. Also, notice that the table is sorted by disk type; although it might be preferable to sort by price, this would result in "Name brand" following "Premium," and again this would violate the rule that cells in the first column of a VLOOKUP range must be sorted. Use Worksheet→Data→Sort on the *entire* table, with the 1st key equal to the cell in the top left corner, if necessary, to put the table in proper order for VLOOKUP.

See also

CHOOSE, HLOOKUP, LOOKUP, MATCH

WEEKDAY

Syntax

WEEKDAY(*serial_number*)

Description

WEEKDAY returns the day of the week of *serial_number*, an integer in the range [1,7], where Sunday is numbered 1, Monday 2, and so on.

You may use a textual date rather than a number; Eureka converts the text to a serial number automatically. However, text in references and arrays is *not* converted and generates a #VALUE! error when used with WEEKDAY.

Example

`WEEKDAY("22-Sep-92")` equals 3.

`TEXT("22-Sep-92","dddd")` equals "Tuesday".

See also

DAY, MONTH, YEAR, HOUR, MINUTE, SECOND

YEAR

Syntax

YEAR(*serial_number*)

Description

YEAR returns the year of *serial_number*, an integer falling in the range 1900-2222.

You may use a textual date rather than a number; Eureka converts the text to a serial number automatically. However, text in references and arrays is *not* converted and generates a #VALUE! error when used with YEAR.

Example

`YEAR("22-Sep-92")` equals 1992.

`YEAR(33000.34)` equals 1990.

See also

DAY, MONTH, HOUR, MINUTE, SECOND

Function Macros

Overview

Function macros allow you to extend Eureka's set of worksheet functions to include ones of your own design. You create a function macro on a separate macro sheet, which is similar to a worksheet, using the familiar worksheet functions and formula entry methods. In addition, Eureka's macro programming language provides several structured programming facilities, such as FOR-NEXT loops and multi-line IF constructs, and programming in the macro language is not unlike programming in languages such as Pascal, C, and modern dialects of BASIC.

Creating a Function Macro

You create a function macro on a separate macro sheet, which is similar to a worksheet. To open a new macro sheet, choose Eureka→New document→Macro sheet.

The general structure of a function macro is illustrated below:

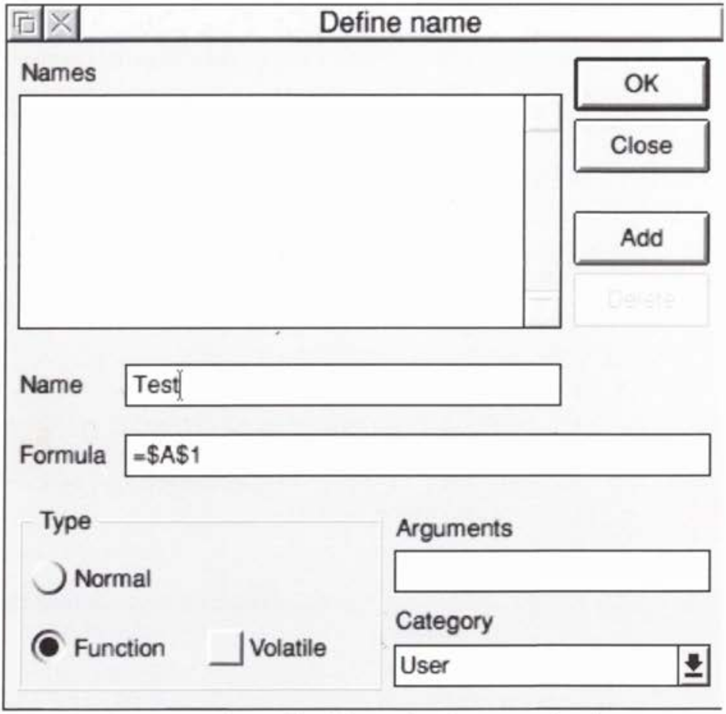
	A	B
1	**Test**	Called as Macro1!Test(arg1,arg2)
2	=ARGUMENT("Arg1")	Assign 1st argument to name Arg1
3	=ARGUMENT("Arg2")	Assign 2nd argument to name Arg2
4	=RESULT(1)	Macro returns numbers only
5	Macro instructions	
6	.	
7	.	
8	.	Should compute something!
9	=RETURN(A8)	Return the value of A8

A formula cell in a macro is called a *statement*. In the example, the ARGUMENT statements create names to hold the function macro's arguments. The macro refers to these names when performing its calculations. The RETURN statement terminates the macro and returns a value to the macro's caller; this value can be any one of Eureka's data types, including references and arrays. However, to return a reference or array, you have to use the RESULT function.

Besides statements (formula cells), you can also create numeric and text cells in a macro sheet. The macro can then refer to these constant cells as required for its computations. Text cells are especially useful for entering comments, which describe to others (and yourself!) how your macro works. Eureka ignores these non-formula cells when running a macro, and you can intersperse them with statements as needed.

Naming a Function Macro

A macro must be named to be used in a formula, and it's useful to enter the name as the first cell in the macro. After entering the name on the macro sheet, you choose Worksheet→Formula→Define name, and the Define name dialogue box appears, as shown below:



Notice that in a macro sheet, this dialogue box has several options not available when used in a worksheet.

- The *Type* option allows you to indicate whether a name is to be a normal, worksheet-style name, or a function macro. When Eureka encounters a normal name while evaluating a formula, it simply evaluates the formula associated with the name. However, when Eureka evaluates a function macro, it runs the program indicated by the name. Thus, in the Define Name dialogue, you should select the *Function* option when creating a name for a function macro.
- The *Volatile* option is available only when the *Function* button is

selected. You would select this option when you want worksheet formulas using the function macro to be included in every worksheet recalculation. This allows you to create function macros which behave like the NOW and RAND worksheet functions, which return different values each time they are used. Formulas using such functions should be included in every worksheet calculation, since they can never be considered "up to date."

- The *Arguments* editable field and *Category* drop-down list also are available only when the Function button is selected. These items allow you to control the function argument list displayed in the Paste Function dialogue box and the function category the macro appears in. Function macros always appear in the User and All categories, but you can also cause them to appear in one of the other categories. Function macro names appear below the built-in Eureka functions in the category lists.

Macro Arguments

Like worksheet functions, macros can accept arguments. You define macro arguments by using the ARGUMENT function, which can assign an argument to a name on the macro sheet, or fill a range on the macro sheet with the argument. The ARGUMENT functions should appear at the beginning of the macro, in the order you write them when calling the macro. For example, if you have defined a macro to be called as Macro!Test(arg1,arg2,arg3), the beginning of the macro *Test* would be similar to the following:

	A
1	Test
2	=ARGUMENT("Arg1")
3	=ARGUMENT("Arg2")
4	=ARGUMENT("Arg3")

The first argument passed to *Test* would be assigned to the name *Arg1*, the second to *Arg2*, and the third to *Arg3*. The macro can then refer to the arguments by using these names. If you omit arguments or supply fewer arguments to a macro than there are ARGUMENT statements, the missing arguments are considered #N/A error values, and Eureka assigns #N/A values to the argument names. If you supply more arguments than there are ARGUMENT statements, Eureka ignores the extra arguments.

See the ARGUMENT function description for more information on controlling the permissible data types of arguments and assigning the argument value to a cell or range of cells, rather than to a name.

Macro Variables

In addition to accepting arguments, macros often require manipulation of intermediate values, called variables, defined within the macro itself. Similarly to its treatment of arguments, Eureka supports two styles of macro variables, names and references. You create a named variable by using the SET.NAME function or the assignment operator, :=, as illustrated below:

	A	B
1	=SET.NAME("i",1)	=i:=1
2	=SET.NAME("Beverage","Cola")	=Beverage:="Cola"
3	=SET.NAME("Array",{1,2,4,8})	=Array:={1,2,4,8}

The statements in column A use the SET.NAME method, while those in column B use the assignment operator, which is just a short-hand for SET.NAME. Thus, the statement in A1 creates a name *i* equal to 1, as does the statement in B1, and you can then use *i* in other formulas. After you run the macro, you can consult the Define Name dialog to see the current values of the names. The major difference between SET.NAME and := is that SET.NAME requires the name in quotes, while := requires the name without quotes.

You can also use a cell or range on the macro sheet as a variable. You can compute a result in a cell, using worksheet functions, and refer to that value elsewhere in the macro sheet, provided the cell has already been calculated during the course of the macro's execution (see the section *Calculation in Macro Sheets* for more information). You can also use the SET.VALUE function to assign a value to a cell on the macro sheet from another cell on the macro sheet. For example, the *TestFact* macro, presented in the FOR-NEXT example in the *Looping in Macros* section, uses SET.VALUE to assign an initial value to a cell used within a FOR-NEXT loop.

Controlling Macro Results

A function macro returns a result to its caller through the RETURN function, and this result can be any one of Eureka's basic data types. By default, a macro can return a number, text string, or logical value. To return a reference or array, or otherwise restrict the result to be a specific type, you have to use the RESULT function to specify the result type. See the RESULT function description for more information on controlling the macro result type.

Decision Making in Macros

In worksheets, the IF function takes two or three arguments and returns a value based on the verity of a logical test. In macro sheets, you can use the IF function in the same way, but you can also use a single argument form, which allows you to select execution of statements spanning multiple rows in the macro sheet. This single argument form, along with the ELSE.IF, ELSE, and END.IF functions, allow you to build flexible conditional structures into your macros.

For example, the following macro returns a text value indicating whether its argument is positive, negative, or zero:

	A
1	Test
2	=ARGUMENT("Num")
3	=IF(Num>0)
4	= RETURN("Positive")
5	=ELSE.IF(Num<0)
6	= RETURN("Negative")
7	=ELSE()
8	= RETURN("Zero")
9	=END.IF()

Although simple, this macro illustrates the basic concepts:

- ELSE.IF is used following IF, when you have several related comparisons but different actions to take on each.
- ELSE provides for a default or "catch-all" action, which is taken when the IF statement and all ELSE.IF alternatives evaluate FALSE. If present, it must appear after any ELSE.IF statements.
- The IF structure is terminated by an END.IF statement. All IF structures must have a matching END.IF.
- Related IF, ELSE.IF, ELSE, and END.IF statements must appear in the same column, and in that relative order, from the lower-numbered rows to the higher.
- Indentation can be used following the opening equals sign to indicate which statements "belong" to others. For example, execution of the statement in cell A4 is controlled by the result of the IF statement in A3, and thus can be thought of as belonging to the IF statement. Statements which belong to another in this way are called *blocks*. Indentation is ignored when running a macro.

Only one statement block in an IF structure is ever selected for execution.

Once it is executed, Eureka proceeds to the cell below the terminal END.IF.

While the statement blocks in the example are only one cell each, they can also consist of multiple cells. Blocks may themselves contain IF structures, as long as they are properly formed.

Looping in Macros

Eureka supports three different types of loops, all of which repeat a set of actions. FOR-NEXT loops are useful when you need to perform an action a specific number of times. WHILE-NEXT and REPEAT-UNTIL loops are useful when you want to perform an action an indefinite number of times, the major difference being that a WHILE loop tests its condition before executing its actions, and a REPEAT loop tests its condition after executing its actions. Thus, a REPEAT loop will execute its actions at least once.

FOR-NEXT Loops

The FOR-NEXT loop is controlled by a *counter variable*, which you specify in the FOR function, along with its initial value, final value, and optionally, a step value, which is added to the counter variable after each loop iteration. For example, the following macro computes the factorial of a number:

	A	B
1	**TestFact**	
2	=ARGUMENT("Num",1)	Set Num, allowing only numbers
3	=SET.VALUE(A5,1)	Initialise the macro result to 1
4	=FOR("i",2,Num)	Loop for i = 2 to Num
5	= A5*i	Multiply the result by i
6	=NEXT()	Next iteration
7	=RETURN(A5)	Return the factorial of Num

In this macro, *i* is the counter variable. The FOR function sets it equal to 2 before comparing it to *Num*. As long as *i* is less than or equal to *Num*, Eureka executes the statement block between the FOR statement and its matching NEXT statement. Because the optional fourth argument to FOR, the step value, is omitted, it is considered equal to 1. Thus, this loop executes its statement block for values of *i* of 2, 3, 4, ... *Num*. If *i* starts out greater than *Num*, the block is not executed, and Eureka proceeds to cell A7, the statement following the terminal NEXT statement.

While the statement block in the example contains only one cell, it can contain as many cells as needed, and it can employ IF constructs and other loop structures.

The *TestFact* macro also illustrates an important use of the SET.VALUE function and circular references in macro sheets. In cell A3, SET.VALUE initialises A5 to 1, without destroying its formula, which contains a direct circular reference. Cell A5 acts much like a variable in other programming languages. We could have achieved the same result using names, as shown below:

	A	B
1	**TestFact**	
2	=ARGUMENT("Num",1)	Set Num, allowing only numbers
3	=FactNum:=1	Initialise the macro result to 1
4	=FOR("i",2,Num)	Loop for i = 2 to Num
5	= FactNum:=FactNum*i	Multiply the result by i
6	=NEXT()	Next iteration
7	=RETURN(FactNum)	Return the factorial of Num

Here, *FactNum* is a name that serves the same purpose as A5 in the prior example. While we could have used the SET.NAME function to define and modify *FactNum*, it is generally clearer to use the name assignment operator, :=. However, if you wish to create or modify a name's value within a function, such as IF, you must use SET.NAME, as the assignment operator cannot be used in a function argument.

WHILE-NEXT Loops

The WHILE-NEXT loop supports looping an indefinite number of times. The loop condition is tested at the beginning of each loop iteration, and a WHILE-NEXT loop will not be executed if the condition initially evaluates FALSE.

The general form of a WHILE-NEXT loop is illustrated below:

	A
1	=WHILE(Some.Condition.Is.True)
2	...Statements
3	=NEXT()

REPEAT-UNTIL Loops

The REPEAT-UNTIL loop supports looping an indefinite number of times. Unlike the WHILE-NEXT loop, the loop condition is tested at the end of the loop, and a REPEAT-UNTIL loop is executed at least once. Also, the WHILE-NEXT loop executes as long as its condition is TRUE, while the REPEAT-UNTIL loop executes until its condition becomes

TRUE.

The general form of a REPEAT-UNTIL loop is illustrated below:

	A
1	=REPEAT()
2	...Statements
3	=UNTIL(Some.Condition.Is.True)

Calculation in Macro Sheets

A major difference between worksheets and macro sheets lies in the calculation method. When Eureka evaluates a formula cell in a worksheet, it may first have to recalculate cells that formula refers to, and once finished with the cell, it will recalculate any other cells which depend on it, if the cell's value changed. This *natural order* method ensures formulas use correct values, and that cells are up to date. However, this method is inappropriate for macro sheets, which implement programs, because programs have a well-defined flow of execution.

In a macro sheet, Eureka evaluates cells sequentially, beginning with the first cell of the macro. It then moves down a row in the same column, calculating the next cell, and so on, until the macro terminates. For example, if a macro is defined in the range A1:A10, Eureka evaluates A1, A2, A3, … A10. Eureka deviates from this strict, columnar, row-by-row order only at your command, implicitly when you use the looping functions and multi-line IF constructs and explicitly when you use GOTO.

Moreover, when you refer to a cell on a macro sheet from a formula also on a macro sheet, Eureka does *not* recalculate the precedent cell. It simply uses its current value. If the precedent cell has not been recalculated or assigned a value during the course of the macro's execution, your formula may use the wrong value. Generally, this means you should only refer to formula cells *above* another formula cell, both in the same macro. However, if you need to make a forward reference, you can initialise a formula cell to a specific value, without destroying the formula, using the SET.VALUE function, as in the *TestFact* macro example presented in the FOR-NEXT discussion earlier in this section.

Because Eureka never attempts to recalculate a macro sheet cell referred to by another macro sheet cell before using its value, circular references are not a problem on macro sheets, and Eureka does not report them in the Status Bar. In fact, circular references can be useful in macro sheets, particularly in loops, where you may want to maintain a variable whose current value depends on its value during the previous loop iteration. You would use SET.VALUE before the loop is entered to assign an initial value to a cell within the loop, while the cell would contain a formula referring to itself. The *TestFact* macro example presented in the FOR-NEXT discussion earlier in this section uses this technique.

Also, when macro sheet cell values change, Eureka will *not* automatically recalculate cells dependent on them. This only presents a problem when you are modifying a macro, and a worksheet using the macro is open. To update the worksheet, you have to recalculate the cells using the macro, but Eureka does not consider those cells in need of calculation. You can deal with this in several ways:

- You can press Shift-F9, which forces recalculation of *every* formula in the worksheet, including those using the function macro. This is the simplest method.
- You can close the dependent worksheet and load it again; Eureka will recalculate cells referring to function macros.
- You can locate the cells needing recalculation, click in the formula bar, entering Edit mode, and then click the Accept Formula button or press Return. Eureka will recalculate the cell. You can also perform a Search/Replace on the "=" character, which begins every formula. That is, the Search and Replace text would be the same, and you would replace in those cells you want to recalculate. Eureka will recalculate all cells "modified" in this manner.

Of course, if function macros failed to recalculate *worksheet* cells before using their values, they would differ seriously from built-in functions. Thus, Eureka *will* recalculate a worksheet cell, if necessary, before using its value in a macro sheet formula.

ARGUMENT

Syntax

ARGUMENT(*name_text,type_num,reference*)

Description

ARGUMENT allows macros to accept arguments, much like the built-in worksheet functions. A macro can accept as many as 14 arguments, and the ARGUMENT functions should appear at the beginning of the macro, in the order you write them in the function.

Name_text is the name on the macro sheet you wish to assign the argument's value, and it should be a valid name enclosed in quotes. If you omit *name_text,* you must specify *reference*.

Type_num indicates the data types that are acceptable for the argument, as shown below.

If *type_num* is	The argument can be
1	Number
2	Text
4	Logical value
8	Reference
16	Error value
64	Array

You can also set *type_num* to the sum of as many of these values as necessary, allowing a function to accept arguments of various types. If you do not specify *type_num,* it is implicitly taken to be equal to 1+2+4 or 7, allowing the argument to be a number, text, or logical value. If you pass an argument that does not match the indicated type, and Eureka cannot convert it to the indicated type, the macro immediately returns the #VALUE! error value.

Reference is a name or reference to a cell or selection on the macro sheet. When you specify *reference,* Eureka assigns the argument value to the indicated cell or selection. *Reference* is very useful when a function macro is to take an array argument, as you can assign the array contents to a cell range. This allows you to modify the array members using the SET.VALUE function, something which is not possible if you assign the array to a name, by specifying *name_text* without also specifying *reference*.

Note: if you specify both *name_text* and *reference*, Eureka assigns *reference* as the value of the name it creates, rather than assigning the argument value to the name. This allows you to refer to the reference using the name. The name may refer to a selection *smaller* than indicated by *reference*, because Eureka will adjust *reference* to reflect the size of the array argument. For example, if you use B:B for *reference*, and the argument is a 10 cell vertical array, then the created name's value will be the reference B1:B10. You can then use the ROWS function to discover the size of the array argument, as in ROWS(Argument.Name), which returns 10. Eureka will *not*, however, expand *reference* to accommodate an array argument that will not fit in *reference*. Instead, it will return the #VALUE! error value immediately, without further processing the macro.

Example

`=ARGUMENT("TextOrNum",1+2)` defines an argument *TextOrNum* which can be text or a number.

`=ARGUMENT("Array",64,C:IV)` defines an argument which can be an array. It uses the *reference* argument to indicate that the array should be placed in the range C:IV, which is equivalent to C1:IV16384. If the argument is a 2 × 3 array, ARGUMENT places the array in the range C1:E2, and sets the name *Array* equal to this reference.

See also

RESULT, SET.NAME, SET.VALUE

BREAK

Syntax

BREAK()

Description

BREAK halts execution of the inner-most loop it appears in. Macro execution resumes at the cell following the NEXT matching a prior FOR or WHILE function, or after the UNTIL matching a prior REPEAT.

Always use BREAK to exit a loop and continue processing the macro, rather than attempting to GOTO a cell outside the loop; generally, use of GOTO for this purpose is not reliable.

See also

CONTINUE, FOR, NEXT, REPEAT, UNTIL, WHILE

CONTINUE

Syntax

CONTINUE()

Description

CONTINUE halts execution of the inner-most loop it appears in, causing execution to resume at the NEXT matching a prior FOR or WHILE function or the UNTIL matching a prior REPEAT. In FOR loops, this implies the counter variable *is* incremented by the step value after using CONTINUE. The loop condition is tested following CONTINUE in all loop types.

Use CONTINUE in preference to GOTO(end_of_loop).

See also

BREAK, FOR, NEXT, REPEAT, UNTIL, WHILE

ELSE

Syntax

ELSE()

Description

ELSE introduces an "else" block in a multi-line IF construct. ELSE must appear alone in a cell below and in the same column as its matching IF. If the matching IF and ELSE.IF logical tests all evaluate FALSE, then Eureka executes the statements between the ELSE and subsequent END.IF statements.

ELSE can only legally appear below a matching IF or ELSE.IF statement and must be matched by an END.IF.

See also

IF, ELSE.IF, END.IF

ELSE.IF

Syntax

ELSE.IF(*logical_test*)

Description

ELSE.IF introduces an "else if" block in a multi-line IF construct. ELSE.IF must appear alone in a cell below and in the same column as its matching IF. If the matching IF's logical test evaluates FALSE, then

Eureka sequentially evaluates the ELSE.IF statements. When Eureka finds an ELSE.IF whose *logical_test* evaluates TRUE, it executes the statements following it, up to a subsequent matching ELSE.IF, ELSE, or END.IF statement. At that point, Eureka proceeds to the cell below the terminal END.IF.

If the IF statement and all the ELSE.IF statements evaluate FALSE, Eureka evaluates the ELSE block, if present.

ELSE.IF can only legally appear below a matching IF statement.

See also

IF, ELSE, END.IF

END.IF

Syntax

END.IF()

Description

END.IF terminates a multi-line IF construct. END.IF must appear alone in a cell below and in the same column as its matching IF.

END.IF can only legally appear below a matching IF, ELSE.IF, or ELSE statement.

See also

IF, ELSE.IF, ELSE

FOR

Syntax

FOR(*counter_text,start_num,end_num,*}*step_num*)**

Description

The FOR function introduces a FOR-NEXT loop, which consists of a FOR statement followed by zero or more statements (the loop body) and terminated by a NEXT statement, where a statement is simply a formula cell. A FOR loop repeats the loop body until the counter variable reaches a specified value.

Counter_text is a name for the counter variable in double quotes.

Start_num is the initial value for the counter variable.

End_num is the final value for the counter variable. Eureka compares the counter variable to *end_num* in each loop iteration in order to decide whether to execute the loop body (see below).

CONTINUE

Syntax

CONTINUE()

Description

CONTINUE halts execution of the inner-most loop it appears in, causing execution to resume at the NEXT matching a prior FOR or WHILE function or the UNTIL matching a prior REPEAT. In FOR loops, this implies the counter variable *is* incremented by the step value after using CONTINUE. The loop condition is tested following CONTINUE in all loop types.

Use CONTINUE in preference to GOTO(end_of_loop).

See also

BREAK, FOR, NEXT, REPEAT, UNTIL, WHILE

ELSE

Syntax

ELSE()

Description

ELSE introduces an "else" block in a multi-line IF construct. ELSE must appear alone in a cell below and in the same column as its matching IF. If the matching IF and ELSE.IF logical tests all evaluate FALSE, then Eureka executes the statements between the ELSE and subsequent END.IF statements.

ELSE can only legally appear below a matching IF or ELSE.IF statement and must be matched by an END.IF.

See also

IF, ELSE.IF, END.IF

ELSE.IF

Syntax

ELSE.IF(*logical_test*)

Description

ELSE.IF introduces an "else if" block in a multi-line IF construct. ELSE.IF must appear alone in a cell below and in the same column as its matching IF. If the matching IF's logical test evaluates FALSE, then

Eureka sequentially evaluates the ELSE.IF statements. When Eureka finds an ELSE.IF whose *logical_test* evaluates TRUE, it executes the statements following it, up to a subsequent matching ELSE.IF, ELSE, or END.IF statement. At that point, Eureka proceeds to the cell below the terminal END.IF.

If the IF statement and all the ELSE.IF statements evaluate FALSE, Eureka evaluates the ELSE block, if present.

ELSE.IF can only legally appear below a matching IF statement.

See also

IF, ELSE, END.IF

END.IF

Syntax

END.IF()

Description

END.IF terminates a multi-line IF construct. END.IF must appear alone in a cell below and in the same column as its matching IF.

END.IF can only legally appear below a matching IF, ELSE.IF, or ELSE statement.

See also

IF, ELSE.IF, ELSE

FOR

Syntax

FOR(*counter_text,start_num,end_num,*__*step_num*__)

Description

The FOR function introduces a FOR-NEXT loop, which consists of a FOR statement followed by zero or more statements (the loop body) and terminated by a NEXT statement, where a statement is simply a formula cell. A FOR loop repeats the loop body until the counter variable reaches a specified value.

Counter_text is a name for the counter variable in double quotes.

Start_num is the initial value for the counter variable.

End_num is the final value for the counter variable. Eureka compares the counter variable to *end_num* in each loop iteration in order to decide whether to execute the loop body (see below).

Step_num is an optional value indicating the amount to add to the counter variable following each iteration of the loop. If you omit *step_num*, Eureka assigns it the value 1.

When processing a FOR loop, Eureka follows these steps:

1. Eureka creates a name *counter_text* and assigns it the value *start_num*. This establishes the counter variable.
2. Eureka compares the counter variable to *end_num*. If the counter variable is less than or equal to *end_num*, assuming a positive *step_num*, Eureka executes the statements up to the matching NEXT. Otherwise, execution proceeds with the statement following the NEXT. If *step_num* is negative, then the loop is entered provided the counter variable is greater than or equal to *end_num*, because the loop is "counting down" to *end_num*.
3. When the NEXT statement is reached, Eureka adds *step_num* to the counter variable and proceeds to step 2.

You can modify the behaviour of a FOR loop in several ways.

- You can exit the loop prematurely via the BREAK function, meaning that Eureka will proceed to the statement following the matching NEXT.
- You can cause Eureka to proceed directly to the matching NEXT (step 3 above) by using the CONTINUE function, skipping any intervening statements.
- You can return from the macro using the RETURN function.
- You can modify the *end_num* or *step_value* within the loop, provided these are references or names, because Eureka evaluates them at each loop iteration.

However, you cannot reliably use the GOTO function to enter or exit any type of loop, including FOR loops.

Example

The following macro uses a FOR-NEXT loop to evaluate the polynomial given by:

$$y = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

The first argument to *PolyEval* is the value x, and the second argument is a horizontal array of coefficients $\{a_n, a_{n-1}, \ldots a_0\}$. For example, PolyEval(1.25,{3,2,4}) returns 11.1875.

	A	B
1	**PolyEval**	
2	=ARGUMENT("X",1)	X-value
3	=ARGUMENT("Coeff",64)	Horizontal array of coeffs
4	=Degree:=COLUMNS(Coeff)-1	Get poly degree
5	=j:=1	Counter for coeff index
6	=SET.VALUE(A8,0)	Initialize the result
7	=FOR("i",Degree,0,-1)	Count down from Degree to 0
8	= A8+INDEX(Coeff,1,j)*X^i	Sum this term
9	= j:=j+1	Next coeff
10	=NEXT()	
11	=RETURN(A8)	

To make this macro more robust, you would add a statement verifying that the array *Coeff* is a horizontal array; for example:

```
IF(ROWS(Coeff)<>1,RETURN(#NUM!))
```

See also

BREAK, CONTINUE, NEXT, REPEAT, UNTIL, WHILE

GOTO

Syntax

GOTO(*reference*)

Description

GOTO causes macro execution to resume at *reference*, which must be a cell reference or name referring to a cell on a macro sheet. Liberal use of GOTO can make the logic of a macro difficult to follow, and thus it is best used sparingly. You can often avoid using GOTO by considering multi-line IF, FOR-NEXT, REPEAT-UNTIL, and WHILE-NEXT constructs.

In addition to the above caveats, there are two restrictions on the use of GOTO:

- You cannot reliably enter or exit a loop with GOTO.
- You cannot reliably enter or exit a multi-line IF construct with GOTO.

Example

`=IF(Variable>10,GOTO(A15))` causes macro execution to continue at cell A15 if *Variable* is greater than 10.

See also

IF, FOR, REPEAT, WHILE, RETURN

IF

Syntax

IF(*logical_test*)

Description

Macro sheets support an additional form of the IF function used in worksheets. The single-argument IF function provides for conditional execution of instructions spanning one or more cells in the macro sheet, while the worksheet-compatible, multi-argument IF function can only return the value of one of its arguments. All IF statements must have a matching END.IF statement, possibly with intervening ELSE.IF and ELSE statements, and even other entire IF blocks.

If the *logical_test* evaluates to TRUE, then Eureka executes the instructions between the IF statement and the matching ELSE.IF, ELSE, or END.IF statement. Once the matching statement is reached, Eureka proceeds to the cell below the matching END.IF.

If *logical_test* evaluates FALSE, then Eureka skips the block of instructions belonging to the IF statement. If ELSE.IF statements are present, Eureka sequentially evaluates them until one evaluates TRUE, at which point the block it begins is executed. If the IF function and all matching ELSE.IF functions evaluate FALSE, Eureka executes the block begun by the matching ELSE statement, if any.

Note that all ELSE.IF, ELSE, and END.IF statements must appear below and in the same column as their matching IF statement. Also, you cannot reliably use GOTO to enter or exit an IF, ELSE.IF, or ELSE block.

Example

The following macro returns the price-per-unit of a product, where the unit price depends on the size of the order. It illustrates all the IF-related functions, as well as nested IF statements. The indentation emphasises the relationships between the IF, ELSE.IF, and ELSE statements and their associated blocks, but is ignored during calculation.

	A
1	**GetUnitPrice**
2	=ARGUMENT("nUnits",1)
3	=IF(nUnits>=100)
4	= RETURN(0.75)
5	=ELSE.IF(nUnits>=50)
6	= RETURN(0.9)
7	=ELSE()
8	= IF(nUnits<0)
9	= RETURN(#NUM!)
10	= ELSE()
11	= RETURN(1.1)
12	= END.IF()
13	=END.IF()

See also

ELSE.IF, ELSE, END.IF

NEXT

Syntax

NEXT()

Description

NEXT terminates a FOR or WHILE loop. All FOR and WHILE functions must have a matching NEXT function, which must appear by itself in a cell below and in the same column as the loop function. In FOR loops, NEXT first causes the counter variable to be incremented by the step value. In both FOR and WHILE loops, NEXT causes execution to resume with the "logical test" in the FOR or WHILE function.

See also

FOR, WHILE

REPEAT

Syntax

REPEAT()

Description

The REPEAT function introduces a REPEAT-UNTIL loop, which consists of a REPEAT statement followed by zero or more statements (the loop body) and terminated by an UNTIL statement, where a statement is simply a formula cell. A REPEAT loop repeats the loop body until the logical test specified in the matching UNTIL function becomes TRUE. Because the REPEAT function makes no test of its own, the loop body is executed at least once.

You can modify the behaviour of a REPEAT loop in several ways.

- You can exit the loop prematurely via the BREAK function, meaning that Eureka will proceed to the statement following the matching UNTIL.
- You can cause Eureka to proceed directly to the matching UNTIL by using the CONTINUE function, skipping any intervening statements.
- You can return from the macro using the RETURN function.
- You can modify the *logical_test* within the UNTIL function, provided it is or uses a reference or name, because Eureka evaluates it at each loop iteration.

However, you cannot reliably use the GOTO function to enter or exit any type of loop, including REPEAT-UNTIL loops.

Example

The infinite geometric series is defined as:

$$y = \sum_{i=0}^{\infty} r^i = \frac{1}{1-r}, \quad -1 < r < 1$$

and converges for $-1 < r < 1$. The macro below determines how many terms are necessary for the result of the series expansion to deviate from the actual value by less than 1 part in 10^6.

	A	B
1	**Geometric**	
2	=ARGUMENT("r")	Must satisfy -1 < r < 1
3	=SET.VALUE(A8,1)	Initialise the series result
4	=SET.VALUE(A9,r)	A9 tracks the powers of r
5	=i:=1	i counts the number of terms
6	=y:=1/(1-r)	y = true value of the series
7	=REPEAT()	Do at least once!
8	= A8+A9	Sum the current term
9	= A9*r	Compute the next term
10	= i:=i+1	Count the term
11	= ABS((A8-y)/y)	Get relative error
12	=UNTIL(A11<1E-6)	Tolerance is 1E-6
13	=RETURN(i)	Return # of terms

For example, Geometric(.1) equals 7, Geometric(.5) equals 20, Geometric(.9) equals 132, and Geometric(.98) equals 684. You need to do two things to make this macro robust:

- You should include a test to ensure that *r* falls within the range (-1,1).
- You should set a maximum value for *i*, so that the loop will terminate after a reasonable number of iterations! You could include a test of *i* against this maximum value in the UNTIL statement, or you might choose to use something like =IF(i>MaxVal,BREAK()) after the statement which increments *i*.

See also

BREAK, CONTINUE, FOR, NEXT, UNTIL, WHILE

RESULT

Syntax

RESULT(*result_type***)**

Description

RESULT allows you to specify the type of value a macro can return through the RETURN function and is required if your macro is to return an array or reference. If necessary, data types are coerced to the type indicated by RESULT. If this coercion is not possible, RETURN returns #VALUE! to its caller.

Result_type should be one of the following values.

If *result_type* is	The macro can return
1	Numbers
2	Text
4	Logical values
8	References
16	Error values
64	Arrays

You can also set *result_type* to the sum of as many of these values as necessary, allowing a function to return any of several different types. If you do not use RESULT in a macro, the result type of the macro is implicitly taken to be equal to 1+2+4 or 7, allowing a macro to return numbers, text, and logical values.

Example

`=RESULT(1+2)` allows a function to return a number or text value.

`=RESULT(8+64)` allows a function to return a reference or array.

See also

RETURN

RETURN

Syntax

RETURN(*return_value*)

Description

RETURN halts execution of a macro, optionally returning a value to its caller. If you do not specify *return_value*, RETURN returns #N/A to its caller.

By default, *return_value* can be a number, text, or logical value. To return an array or reference, specify the macro return type by using the RESULT function.

Example

`=RETURN(D1)` returns the *value* of cell D1 on the macro sheet. To return the *reference* D1, use `=RESULT(8)` beforehand.

See also

RESULT

UNTIL

Syntax

UNTIL(*logical_test*)

Description

UNTIL terminates a REPEAT loop. All REPEAT functions must have a matching UNTIL function, which must appear by itself in a cell below and in the same column as the REPEAT function. The UNTIL function causes execution to resume with the cell containing its matching REPEAT function, if *logical_test* evaluates FALSE. If *logical_test* evaluates TRUE, execution resumes at the cell immediately below the cell containing the UNTIL function.

See also

REPEAT

SET.NAME

Syntax

SET.NAME(*name_text*,*value*)

Description

SET.NAME allows you to create a new name on the macro sheet and assign it a value. The name created is global to the macro sheet; that is, it can be accessed and modified outside the macro that created it.

Name_text is the name you wish to create in double quotes.

Value is the value you wish to assign to the name. It will typically be a constant value, including references and arrays, but it also can be any valid Eureka formula. However, if *value* is a formula, it will be evaluated only once, when the SET.NAME function creates the name. That is, the formula is evaluated, and its value at the time of the SET.NAME function call is assigned to the name, as a constant.

If you omit *value*, Eureka deletes the name specified by *name_text*.

Example

`=SET.NAME("Counter",1)` assigns the value 1 to the name *Counter*. We could also have used the assignment operator, :=, as follows:

```
=Counter:=1.
```

This notation is more compact than SET.NAME and has the same effect. However, you must use SET.NAME to delete a name from the macro sheet, or if you want to define a name within a function call. For example,

```
=IF(Value<10,SET.NAME(Value,10))
```

ensures *Value* is at least 10.

See also

ARGUMENT, SET.VALUE

SET.VALUE

Syntax

SET.VALUE(*reference*,*value*)

Description

SET.VALUE allows you to set cell values on a macro sheet, without destroying any formulas contained in those cells.

Reference is a reference or name indicating the cells you want to change. It can be a single cell or a range.

Value specifies the new value or values for *reference*. If *value* is an array smaller than *reference*, the array is expanded to fill the range, using the normal rules for array expansion. If *value* is an array larger than *reference*, SET.VALUE fills as many cells as possible with the array contents.

Example

The following macro contains two functions, *MZero*, and *MIdentity*, which return the $n \times n$ zero and identity matrices, respectively. SET.VALUE is used to fill a scratch array with 0's in both cases, and to set the diagonal to 1's for *MIdentity*.

	A	B
1	**MZero**	Zero Matrix Function
2	=bZero:=TRUE	Record in Boolean flag
3	=GOTO(A6)	Enter "real" function
4	**MIdentity**	Identity Matrix Function
5	=bZero:=FALSE	Record in Boolean flag
6	=ARGUMENT("nOrder",1)	Matrix order
7	=RESULT(64)	Allow array result
8	=MaxDim:=50	Maximum dimension
9	=IdentRange:=C:IV	Scratch area
10	=nOrder:=INT(nOrder)	nOrder must be an integer
11	=IF(nOrder<2#OR#nOrder>MaxDim, RETURN(#NUM!))	Range check
12	=ref:=INDEX(IdentRange,1,1):INDEX (IdentRange,nOrder,nOrder)	Create nOrder x nOrder scratch area
13	=SET.VALUE(ref,0)	Fill scratch area with zeros
14	=IF(NOT(bZero))	If bZero is FALSE, then we return the identity matrix
15	= FOR("i",1,nOrder)	
16	= SET.VALUE(INDEX(ref,i,i),1)	Set diagonal element to 1
17	= NEXT()	
18	=END.IF()	
19	=RETURN(ref)	Return the array

See also

ARGUMENT, SET.NAME

WHILE

Syntax

WHILE(*logical_test*)

Description

The WHILE function introduces a WHILE-NEXT loop, which consists of a WHILE statement followed by zero or more statements (the loop body) and terminated by a NEXT statement, where a statement is simply a formula cell. A WHILE loop repeats the loop body as long as *logical_test* evaluates TRUE.

You can modify the behaviour of a WHILE loop in several ways.

- You can exit the loop prematurely via the BREAK function, meaning that Eureka will proceed to the statement following the matching NEXT.
- You can cause Eureka to proceed directly to the matching NEXT by using the CONTINUE function, skipping any intervening statements.
- You can return from the macro using the RETURN function.
- You can modify the *logical_test* within the WHILE function, provided it is or uses a reference or name, because Eureka evaluates it at each loop iteration.

However, you cannot reliably use the GOTO function to enter or exit any type of loop, including WHILE-NEXT loops.

Example

Occasionally, one needs to implement an "infinite loop" structure. The example below accomplishes this with WHILE-NEXT. Note the use of the BREAK function to exit the loop when some condition becomes TRUE. We could also have used RETURN to exit the loop.

	A
1	=WHILE(TRUE)
2	Other statements...
3	= IF(WeShouldStopNow,BREAK())
4	Other statements...
5	=NEXT()

See also

BREAK, CONTINUE, FOR, NEXT, REPEAT, UNTIL

Index

H

I

K

L

M

T

U

V

W

Z