DOWNLOAD PDF THE POWER OF VARIABLE NAMES

Chapter 1 : Variables in DAX - SQLBI

Share The content about studying Code complete2 ch11 - the power of variable names Slideshare uses cookies to improve functionality and performance, and to provide you with relevant advertising. If you continue browsing the site, you agree to the use of cookies on this website.

Data Types[edit] PowerShell data types include integers, floating point values, strings, Booleans, and datetime values. GetType Displays String Type Conversion[edit] Variables may be converted from one type to another by explicit conversion. This script demonstrates data types and data type conversion. Quotes[edit] Single quotes display literal strings as is, without interpretation. Double quotes evaluate strings before displaying them. Note the use of both single and double quotes for string processing. This script displays a message based on user input. Get-Variable and Microsoft TechNet: Start a new PowerShell session and use Get-Variable to display a list of all automatic variables and values that are defined by default when a new session is started. Using the Read-Host Cmdlet. Add a comment at the top of the script that describes the purpose of the script. Experiment with the Hello script above using both using single quotes and double quotes to display the strings to ensure that you understand the difference between the two. Review Windows IT Pro: Experiment with entering different types of data integers, floating point values, dates, strings and use GetType to display the data type entered. Experiment with data type conversion type casting and strongly typed variables when entering different types of data. Lesson Summary[edit] A variable or scalar is a storage location and an associated symbolic name an identifier which contains some known or unknown quantity or information, a value. You can assign almost anything to a variable, even complete command results. In Windows PowerShell, single quotes display literal strings as is, without interpretation. Click on a question to see the answer. A variable or scalar is a storage location and an associated symbolic name an identifier which contains some known or unknown quantity or information, a value. A constant is an identifier whose associated value cannot typically be altered by the program during its execution. Many programming languages employ a reserved value often named null or nil to indicate an invalid or uninitialized variable. In almost all languages, variable names cannot start with a digit 0â€”9 and cannot contain whitespace characters. A data type or simply type is a classification identifying one of various types of data, such as real, integer or Boolean, that determines the possible values for that type, the operations that can be done on values of that type; the meaning of the data; and the way values of that type can be stored. Type conversion, typecasting, and coercion are different ways of, implicitly or explicitly, changing an entity of one data type into another. Scope can vary from as little as a single expression to as much as the entire program, with many possible gradations -- such as code block, function, or module -- in between. PowerShell variable names are not case sensitive. PowerShell data types include integers, floating point values, strings, Booleans, and datetime values.

DOWNLOAD PDF THE POWER OF VARIABLE NAMES

Chapter 2 : Understanding and Creating Windows PowerShell Variables - Techotopia

Code Complete 2 - Steve McConnell - The Power of Variable Names I just love Steve McConnell's classic book Code Complete 2, and I recommend it to everyone in the Software 'world' who's willing to progress and sharpen his skills.

The version of the DAX language has many new functions, but none of them is a game changer for the language as variables are. Variables are a major feature that makes writing DAX code easier. Moreover, it greatly increases the readability and reusability of your code. Syntax is very simple, as in the following example: You can have as many variables as needed in a single expression, and each one has its own VAR definition. From the previous example, you can already appreciate the advantage in terms of code readability. By assigning names to expressions, the code is simpler to read and maintain over time. Yet, there is a lot more to learn about variables. First, variables can contain tables. For example, you can define an expression like the following one: You can define variables with both scalar values and tables. There is no difference in the syntax or in their usage. Another point that might not be obvious is that variables are part of an expression. Wherever you write an expression, you can define a variable. Moreover, DAX evaluates variables in the context of their definition, not in the one where they are used. In standard DAX, before variables, you had to write: The result is four products, of different colors: The problem of this formula and all of the formulas that follow the same pattern is that it stops working as soon as you put any filter on the product outside of it, e. By using variables, the query becomes much easier to author and works regardless of outer filters: It is worth spending a few minutes reading the formula, because it shows the power of variables very well. Said in other words, variables let you access the outer filter context, which is probably the single most wanted feature of seasoned DAX coders. You are not limited to use variables in measures or in queries, they work perfectly fine in calculated columns. For example, to count the number of products with a price higher than the current one, you previously had to write a calculated column like this: By using variables, the same expression becomes much clearer and easier to write: Thus, it evaluates the price of the current product. Finally, variables lead to the single evaluation of complex subexpression. The DAX optimizer use variables to guarantee that their evaluation happens only once, resulting in much faster code, whenever you had the same subexpression that needs to be evaluated more than once. Learning DAX from scratch?

Chapter 3 : How to get variable name in PowerShell - Stack Overflow

Stay ahead with the world's most comprehensive technology and business learning platform. With Safari, you learn the way you learn best. Get unlimited access to videos, live online training, learning paths, books, tutorials, and more.

If you have followed my blog for a while, I have a ton of tutorials about percent change. S ee list of other tutorials dealing with percent change Now, for the twist making this calculation more difficult. I want to calculate percent change even when a filter is applied by clicking on a visual. By using variables we can apply some data modeling voodoo magic to get the desired results. See a sample of the measures working in the Power BI Embedded sample below: If you need some more help loading the data follow this tutorial about loading data using the Advanced Query Editor. This tutorial teaches you how to copy and paste M code into the Advanced Editor. Year [Date] , type number in "Inserted Year" Your loaded data should look like the following: It is a good practice to review the data columns and verify that the formatting is set up correctly. Repeat this for the following columns: See the image below for the bar chart set up: Create Bar Chart To start we will create a explicit calculation for the total selected sales. To learn more about explicit vs implicit calculations within Power BI Click the following link. This allows us to see the entire sales number. Make sure you have the Total Selected Sales measure selected by Clicking on the words of the measure. The final format of the card visual should look like the image below: Final Card Format For the next step we will repeat the previous steps for our new measures. Create the measure Change the formatting of the card Change the formatting of the measure The next measure will calculate the prior year total sales, but only calculate this value when a filter context from a different visual is applied. Take note this is the magic I was talking about!! First, we are creating a variable with the VAR. In the below image I show you that the variable that we are defining is called selectedYear. SelectedYear returns a value of blank if multiple years are selected. The Return in then allows us to output a value. The next part is crucial to making this work. Within the Calculate we are applying two filters. The first filter selects the entire table with the All then we apply the filter from the earlier defined variable with the name of selectedYear. Prior Year Sales in Color Apply the same formatting steps 2, and 3 mentioned earlier. Now, we can select one of the years within our bar chart. Doing so populates the two measures we created. The selected year total sales, and the prior year of sales. Selected Bar Now we will make our percent change measure. Enter the following as a new measure: Add a final measure that calculates the difference. Thanks for following along an learning about variables. Trust me, these are super helpful. Make sure you spend some learning how to leverage variables in your DAX equations. I mean, come on, Ferrari, how baller is that last name! I guess that is why he is the master. Kudos to you Alberto! Still need more reading about DAX, check out the following books:

Chapter 4 : Variable (mathematics) - Wikipedia

This Video Summarizes Chapter 11 of the Notorious Book "Code Complete". Steve McConnell's masterpiece is a timeless classic and it's a must-read for all programmers. my endeavor is to summarize.

To distinguish them, the variable x is called an unknown, and the other variables are called parameters or coefficients, or sometimes constants, although this last terminology is incorrect for an equation and should be reserved for the function defined by the left-hand side of this equation. In the context of functions, the term variable refers commonly to the arguments of the functions. This is typically the case in sentences like " function of a real variable ", "x is the variable of the function f: In the same context, variables that are independent of x define constant functions and are therefore called constant. For example, a constant of integration is an arbitrary constant function that is added to a particular antiderivative to obtain the other antiderivatives. Because the strong relationship between polynomials and polynomial function , the term "constant" is often used to denote the coefficients of a polynomial, which are constant functions of the indeterminates. This use of "constant" as an abbreviation of "constant function" must be distinguished from the normal meaning of the word in mathematics. Other specific names for variables are: An unknown is a variable in an equation which has to be solved for. An indeterminate is a symbol, commonly called variable, that appears in a polynomial or a formal power series. Formally speaking, an indeterminate is not a variable, but a constant in the polynomial ring or the ring of formal power series. However, because of the strong relationship between polynomials or power series and the functions that they define, many authors consider indeterminates as a special kind of variables. A parameter is a quantity usually a number which is a part of the input of a problem, and remains constant during the whole solution of this problem. For example, in mechanics the mass and the size of a solid body are parameters for the study of its movement. In computer science , parameter has a different meaning and denotes an argument of a function. Free variables and bound variables A random variable is a kind of variable that is used in probability theory and its applications. It should be emphasized that all these denominations of variables are of semantic nature and that the way of computing with them syntax is the same for all. Dependent and independent variables[edit] Main article: Dependent and independent variables In calculus and its application to physics and other sciences, it is rather common to consider a variable, say y, whose possible values depend on the value of another variable, say x. In mathematical terms, the dependent variable y represents the value of a function of x. To simplify formulas, it is often useful to use the same symbol for the dependent variable y and the function mapping x onto y. For example, the state of a physical system depends on measurable quantities such as the pressure , the temperature , the spatial position, In the formulas describing the system, these quantities are represented by variables which are dependent on the time, and thus considered implicitly as functions of the time. Therefore, in a formula, a dependent variable is a variable that is implicitly a function of another or several other variables. An independent variable is a variable that is not dependent. For example, in the notation f x, y, z , the three variables may be all independent and the notation represents a function of three variables. On the other hand, if y and z depend on x are dependent variables then the notation represents a function of the single independent variable x.

Chapter 5 : Solved: Dynamic column name from its value - Microsoft Power BI Community

The Get-Variable cmdlet gets the PowerShell variables in the current console. You can retrieve just the values of the variables by specifying the ValueOnly parameter, and you can filter the variables returned by name. This command gets variables with names that begin with the letter m. The command.

To assign a value to a variable therefore, the variable name is placed on the left of the expression, followed by the assignment operator. The value to be assigned is then placed to the right of the assignment operator. PowerShell is a loosely typed language. Instead, PowerShell infers the type from the data being assigned. Since "Red" is clearly a string, PowerShell safely assumes the variable is of type string. We can similarly declare a variable to contain an integer value: Once again, the type of the variable can be inferred by PowerShell, in this case an Int32 number. In fact, since all types are essentially objects, the type of a variable may be identified at any time by calling the GetType method of the object: ValueType Having said that PowerShell is a loosely typed language, it is important to note that the data type may be explicitly declared if desired. In practice, accessing a variable is as simple as referencing the name it was given when it was created. For example, if we want to display the value which we assigned to our numberofcolors variable we can simply reference it in a command: The number of colors is 6. Similarly we can display the value of the mycolor variable: We will look at these types in detail in the next chapter Windows PowerShell 1. First we are going to look at changing the type of a variable after it has been created. As previosuly mentioned, PowerShell is what is termed a loosely typed language. This contrasts with programming languages such as Java which are strongly typed languages. The rules of a strongly typed language dictate that once a variable has been declared as a particular type, its type cannot later be changed. In Java, for example, you cannot assign a floating point number to a variable that was initially declared as being a string. Loosely typed languages such as PowerShell and JavaScript , on the other hand, allow the variable type to be changed at any point in the life of the variable simply by assigning a value of a different type to it. For example, we can create a variable, assign it an integer value and later change it to a string type by assigning a string to it: Pre-defined PowerShell Variables In an early section of this chapter the existence of pre-defined Windows PowerShell variables was mentioned specifically, it is not possible to create a variable using a name assigned to a pre-defined variable. The final area to be covered in this chapter, therefore, is to list some of the more useful pre-defined variables:

Chapter 6 : PowerShell/Variables - Wikiversity

NOTE: I do plan on getting the "values" of each variable at a different point in the script, so just need the variable names here. Thanks in advance! variables powershell.

I want to see the Practice Problems for this section. Take me back to the Main Page Powers: Definition and Simple Powers with Numbers Even though you may not be aware of it, you have already encountered the concept of "raising a number to a power". Consider the following simple example: We can re-write this as: This notations simply means that if we take the number 3 and multiply it by itself we get 9. This means that three threes multiplied together gives us In the above examples we say that the number 3 is raised to a power. If 3 is raised to the 2nd power, we get 9 and if 3 is raised to the 3rd power we get In fact, writing the number of times that we multiply something into itself as a power - with the notation of the power as being a superscript to the right of the number - can save us quite a lot of space!. Suppose we were multiplying 3 by itself 6 times, and compare the following: Futhermore, as you will see below, this way of indicating numbers and powers makes the algebraic manipulation of numbers and multiplication of large values much much easier. Why are Powers Relevant? Powers - or rather, the exponential notation introduced above - is extremely important in astronomy. The very basic reason for this is that astronomy forces us to expand our understanding of scales and sizes of objects. You may go back and review the sections on powers of 10 to see what kind of scales we are talking about when we refer to concepts such as "the age of the earth", "the age of the universe", "the size of the sun", etc Let us take the Earth-Sun distance for instance; you can look this up in any astronomy textbook and you will notice a number written as follows: This means that this distance is times a million km note: If you had to write this number out you would write: Now imagine trying to use these number in mathematical relationships to calculate some other value; there is a simple clear advantage to the exponential power notation! There are many occasions also that one is simply interesting in orders of magnitude, and the exact details of a number are not important. Powers with Variables To generalize the notion of powers fruther, we can imagine any number being multiplied by itself any number of times. We call a general - underspecified - number a variable, and the canonical notation for an algebraic variable is x. Thus, we say that x2 is "x raised to the power of two" or equivalently "x squared" which is x mulitplied by itself. In fact, if we multiply the variable x by itself a number of times say n - times we get: The number represented here as x is called the base, and n is called the power or exponent. The basic definition of a number written in exponential notation states that the base should be multiplied by itself the number of times indicated by the exponent. Algebraic Rules for Working with Powers From the above definition of powers some basic algebraic rules follow. Consider the following; suppose we multiply the following: Thus, in multiplying numbers in exponential notation - following the rules of number multiplication - the power must add: This, in fact, generalizes to any base and any power, and we can write it interms of our variable notation introduced above. From this, the division rule for powers is easy to see as well: Another rule that emerges out of the algebra of exponential powers is that if a number raised to a power and is in the denominator of a fraction , then it can be written as "raised to a negative power". For instance, consider the following example: Another important observation about exponential powers rests in the fact that a number raised to some power can itself be raised to a power again! An example can best illustrate this point. According to our definition above - of algebraic powers -we can write this as: Thus, when a number to some power is raised to a power again, we multiply the two powers to get the resultant value: Finally, one more important piece of knowledge about exponents: These are the basic rules for working with and manipulating exponents and powers. You need to keep these in mind and they will show up fequently in making qualitative and quantitave argumetns in astronomy. Here is a summary of what has been mentioned above, in more general notation. A negative exponent indicates that the power is in the denominator: This should give you a basic start on working with and understanding powers and exponents. Remember that in the general notation used for the above rules we write x to mean that for any given number the rule works Same goes with the exponents n, m. We write it in this general notation because it is simpler to state rules this way. Do not let this notation with letter variables

confuse you. When you encounter a problem and you see that you need to take say 5 to the third power and multiply it by 5 to the 10th power, you simply realize that for the given situation: Practicing with problems and applying these rules will make them easier to understand. Take me back to the Main Page.

Chapter 7 : Software Engineering - Code Complete: The Power of Variable Names (showing of 3)

I understand the benefits of using good variable names, but I noticed that short expressions created using short variable names are easier to read than long expressions created with good variable names, especially if a short expression reminds of well known equation.

Variables[edit] A variable is something that holds a value that may change. In simplest terms, a variable is just a box that you can put stuff in. You can use variables to store all kinds of stuff, but for now, we are just going to look at storing numbers in variables. When we ask Python to tell us what is stored in the variable lucky, it returns that number again. We can also change what is inside a variable. Then, we assign the integer 9 to changing, and ask again what is stored in changing. Python has thrown away the 3, and has replaced it with 9. Next, we create a second variable, which we call different, and put 12 in it. Now we have two independent variables, different and changing, that hold different information, i. You can also assign the value of a variable to be the value of another variable. First the name, then the value. We start out declaring that red is 5, and blue is As you can see, you can pass several arguments to print to tell it to print multiple items on one line, separating them by spaces. As expected, Python reports that red stores 5, and blue holds Now we create a third variable, called yellow. To set its value, we tell Python that we want yellow to be whatever red is. Python knows that red is 5, so it also sets yellow to be 5. Python looks up the value of blue, and finds that it is After this assignment Python reports that yellow is 5, red is 10, and blue is The reason that yellow is still 5 when red is 10, is because we only said that yellow should be whatever red is at the moment of the assignment. The interplay between different variables in Python is, in fact, more complex than explained here. The above example works with integer numbers and with all other basic data types built into Python; the behavior of lists and dictionaries you will encounter these complex data types later is entirely different, though. You may read the chapter on data types for a more detailed explanation of what variables really are in Python and how their type affects their behavior. However, it is probably sufficient for now if you just keep this in mind: A character is anything you can type on the keyboard in one keystroke, like a letter, a number, or a backslash. For example, "hello" is a string. It is five characters long â€" h, e, l, l, o. Strings can also have spaces: There are no limits to the number of characters you can have in a string â€" you can have anywhere from one to a million or more. You can even have a string that has 0 characters, which is usually called "the empty string. In all cases, you start and end the string with your chosen string declaration. Placing a backslash directly before another symbol like this is known as escaping the symbol. Note that if you want to put a backslash into the string, you also have to escape the backslash, to tell Python that you want to include the backslash, rather than using it as an escape character. As you can see from the above examples, only the specific character used to quote the string needs to be escaped. This makes for more readable code. Notice that there is a space at the end of the first string. Variables can store much more than just numbers. You can also use them to store strings! Then, we just tell Python to print out whatever is inside the question variable. Notice that when we tell Python to print out question, there are no quotation marks around the word question: If we put in quotation marks around question, Python would treat it as a string, and simply print out question instead of What did you have for lunch?. You can also use a different function called input , which works in nearly the same way. We will learn the differences between these two functions later. That is, the new input function reads a line from sys. It raises EOFError if the input is terminated prematurely e. To get the old behavior of input , use eval input. In this program, we created a variable called answer, and put whatever the user wrote into it. Then, we print out a new string, which contains whatever the user wrote. Notice the extra space at the end of the "You had " string, and the exclamation mark at the start of the "! Traceback most recent call last: Python is telling us that there is a TypeError, which means there is a problem with the types of information being used. For example, Python thinks that the number variable is holding a string, instead of a number. If the user enters 15, then number will contain a string that is two characters long: So how can we tell Python that 15 should be a number, instead of a string? Also, when printing out the answer, we are telling Python to concatenate together a string "If we add 10 to your number, we get " and a number plusTen. How do we tell Python to treat a number as a string, so

that we can print it out with another string? Luckily, there are two functions that are perfect solutions for these problems. The int function will take a string and turn it into an integer, while the str function will take an integer and turn it into a string. In both cases, we put what we want to change inside the parentheses. Therefore, our modified program will look like this: Another way of doing the same is to add a comma after the string part and then the number variable, like this: List of Learned Functions[edit] print: Note that in version 3. This string will be displayed on the prompt while waiting for the user input. This is why using input as a way to get a user string value returns an error because the user needs to enter strings with quotes. This way we do not have to bother with error messages until the error handling chapter and will not make a security vulnerability in your code. Exercises[edit] Write a program that asks the user to type in a string, and then tells the user how long that string was. Ask the user for a string, and then for a number. Print out that string, that many times. For example, if the string is hello and the number is 3 you should print out hellohellohello. What would happen if a mischievous user typed in a word when you ask for a number?

Chapter 8 : Using Variables within DAX - Power BI Tips and Tricks

The PowerShell variable has many features that help you create reliable scripts. Understanding the most important concepts of the PowerShell variable is essential for everything you do in PowerShell. In the first post of this series, I will explain how to name variables, how to assign values, and how to deal with PowerShell data types.

Polynomials of small degree have been given specific names. A polynomial of degree zero is a constant polynomial or simply a constant. Polynomials of degree one, two or three are respectively linear polynomials, quadratic polynomials and cubic polynomials. For higher degrees the specific names are not commonly used, although quartic polynomial for degree four and quintic polynomial for degree five are sometimes used. The names for the degrees may be applied to the polynomial or to its terms. The polynomial 0, which may be considered to have no terms at all, is called the zero polynomial. Unlike other constant polynomials, its degree is not zero. The zero polynomial is also unique in that it is the only polynomial having an infinite number of roots. In the case of polynomials in more than one indeterminate, a polynomial is called homogeneous of degree n if all its non-zero terms have degree n. The zero polynomial is homogeneous, and, as homogeneous polynomial, its degree is undefined. For more details, see homogeneous polynomial. The commutative law of addition can be used to rearrange terms into any preferred order. In polynomials with one indeterminate, the terms are usually ordered according to degree, either in "descending powers of x", with the term of largest degree first, or in "ascending powers of x". The polynomial in the example above is written in descending powers of x. The first term has coefficient 3, indeterminate x, and exponent 2. The third term is a constant. Because the degree of a non-zero polynomial is the largest degree of any one term, this polynomial has degree two. It may happen that this makes the coefficient 0. The term "quadrinomial" is occasionally used for a four-term polynomial. A real polynomial is a polynomial with real coefficients. The argument of the polynomial is not necessarily so restricted, for instance the s-plane variable in Laplace transforms. A real polynomial function is a function from the reals to the reals that is defined by a real polynomial. Similarly, an integer polynomial is a polynomial with integer coefficients, and a complex polynomial is a polynomial with complex coefficients. A polynomial in one indeterminate is called a univariate polynomial, a polynomial in more than one indeterminate is called a multivariate polynomial. A polynomial with two indeterminates is called a bivariate polynomial. These notions refer more to the kind of polynomials one is generally working with than to individual polynomials; for instance when working with univariate polynomials one does not exclude constant polynomials which may result, for instance, from the subtraction of non-constant polynomials , although strictly speaking constant polynomials do not contain any indeterminates at all. It is possible to further classify multivariate polynomials as bivariate, trivariate, and so on, according to the maximum number of indeterminates allowed. Again, so that the set of objects under consideration be closed under subtraction, a study of trivariate polynomials usually allows bivariate polynomials, and so on. It is common, also, to say simply "polynomials in x, y, and z", listing the indeterminates allowed. The evaluation of a polynomial consists of substituting a numerical value to each indeterminate and carrying out the indicated multiplications and additions.

Chapter 9 : Polynomial - Wikipedia

It is worth spending a few minutes reading the formula, because it shows the power of variables very well. The variable OnePercentOfSales is defined inside ADDCOLUMNS. Thus, it is evaluated in the filter context in which ADDCOLUMNS is computed, i.e. that of black products.

Note the semicolon at the end of the line; that is how your compiler separates one program statement from another. Note that we must specify the type of data that a variable will store. Multiple variables can be declared with one statement, like this: In early versions of C, variables had to be declared at the beginning of a block. After declaring variables, you can assign a value to a variable later on using a statement like this: Naming Variables[edit] Variable names in C are made up of letters upper and lower case and digits. Names must not begin with a digit. Some examples of valid but not very descriptive C variable names: It is not allowed to use the same name for multiple variables in the same scope. When working with other developers, you should therefore take steps to avoid using the same name for global variables or function names. Some large projects adhere to naming guidelines [1] to avoid duplicate names and for consistency. In addition there are certain sets of names that, while not language keywords, are reserved for one reason or another. For example, a C compiler might use certain names "behind the scenes", and this might cause problems for a program that attempts to use them. Also, some names are reserved for possible future use in the C standard library. For now, just avoid using names that begin with an underscore character. The naming rules for C variables also apply to naming other language constructs such as function names, struct tags, and macros, all of which will be covered later. Literals[edit] Anytime within a program in which you specify a value explicitly instead of referring to a variable or some other form of data, that value is referred to as a literal. In the initialization example above, 3 is a literal. Literals can either take a form defined by their type more on that soon , or one can use hexadecimal hex notation to directly insert data into a variable regardless of its type. They are int, char, float, and double. The int type[edit] The int type stores integers in the form of "whole numbers". An integer is typically the size of one machine word, which on most modern home PCs is 32 bits 4 octets. Examples of literals are whole numbers integers such as 1,2,3, 10, When int is 32 bits 4 octets , it can store any whole number integer between and A 32 bit word number has the possibility of representing any one number out of possibilities 2 to the power of If you want to declare a new int variable, use the int keyword. The char type[edit] The char type is capable of holding any member of the execution character set. It stores the same kind of data as an int i. In standard C it never can be less than 8 bits. A variable of type char is most often used to store character data, hence its name. Note that the char value must be enclosed within single quotations. When we initialize a character variable, we can do it two ways. One is preferred, the other way is bad programming practice. One important thing to mention is that characters for numerals are represented differently from their corresponding number, i. There is one more kind of literal that needs to be explained in connection with chars: A string is a series of characters, usually intended to be displayed. An example of a string literal is the "Hello, World! The string literal is assigned to a character array, arrays are described later. It stores inexact representations of real numbers, both integer and non-integer values. It can be used with numbers that are much greater than the greatest possible int. It is important to note that floating-point numbers are inexact. Some numbers like 0. Very large and very small numbers will have less precision and arithmetic operations are sometimes not associative or distributive because of a lack of precision. Nonetheless, floating-point numbers are most commonly used for approximating real numbers and operations on them are efficient on modern microprocessors. A float is only one machine word in size. Therefore, it is used when less precision than a double provides is required. The double type[edit] The double and float types are very similar. The float type allows you to store single-precision floating point numbers, while the double keyword allows you to store double-precision floating point numbers â€” real numbers, in other words. Its size is typically two machine words, or 8 bytes on most machines. Examples of double literals are 3. If you use 4 instead of 4. The distinction between floats and doubles was made because of the differing sizes of the two types. When C was first used, space was at a minimum and so the judicious use of a float

instead of a double saved some memory. Nowadays, with memory more freely available, you rarely need to conserve memory like this â€” it may be better to use doubles consistently. Indeed, some C implementations use doubles instead of floats when you declare a float variable. If you want to use a double variable, use the double keyword. For completeness, it is important to mention that sizeof is a unary operator , not a function. Note that when sizeof is applied to a char, the result is 1; that is: Data type modifiers[edit] One can alter the data storage of any data type by preceding it with certain modifiers. The int keyword need not follow the short and long keywords. This is most commonly the case. A short can be used where the values fall within a lesser range than that of an int, typically to A long can be used to contain an extended range of values. It is not guaranteed that a short uses less memory than an int, nor is it guaranteed that a long takes up more memory than an int. Typically a short is 2 bytes, an int is 4 bytes, and a long either 4 or 8 bytes. Modern C compilers also provide long long which is typically an 8 byte integer. In all of the types described above, one bit is used to indicate the sign positive or negative of a value. If you decide that a variable will never hold a negative value, you may use the unsigned modifier to use that one bit for storing other data, effectively doubling the range of values while mandating that those values be positive. The unsigned specifier also may be used without a trailing int, in which case the size defaults to that of an int. There is also a signed modifier which is the opposite, but it is not necessary, except for certain uses of char, and seldom used since all types except char are signed by default. The long modifier can also be used with double to create a long double type. This floating-point type may but is not required to have greater precision than the double type. To use a modifier, just declare a variable with the data type and relevant modifiers: It is then not allowed to be changed. While the idea of a variable that never changes may not seem useful, there are good reasons to use const. For one thing, many compilers can perform some small optimizations on data when it knows that data will never change. Note that a Standard conforming compiler must issue a warning if an attempt is made to change a const variable - but after doing so the compiler is free to ignore the const qualifier. Magic numbers[edit] When you write C programs, you may be tempted to write code that will depend on certain numbers. For example, you may be writing a program for a grocery store. This complex program has thousands upon thousands of lines of code. The programmer decides to represent the cost of a can of corn, currently 99 cents, as a literal throughout the code. Now, assume the cost of a can of corn changes to 89 cents. The programmer must now go in and manually change each entry of 99 cents to While this is not that big of a problem, considering the "global find-replace" function of many text editors, consider another problem: To reliably change the price, you have to look at every occurrence of the number C possesses certain functionality to avoid this. This functionality is approximately equivalent, though one method can be useful in one circumstance, over another. Using the const keyword[edit] The const keyword helps eradicate magic numbers. By declaring a variable const corn at the beginning of a block, a programmer can simply change that const and not have to worry about setting the value elsewhere. There is also another method for avoiding magic numbers. It is much more flexible than const, and also much more problematic in many ways. It also involves the preprocessor, as opposed to the compiler. For some purposes, define can be harmfully used, and it is usually preferable to use const if define is unnecessary. It is possible, for instance, to define, say, a macro DOG as the number 3, but if you try to print the macro, thinking that DOG represents a string that you can show on the screen, the program will have an error. It disregards the structure of your program, replacing the text everywhere in effect, disregarding scope , which could be advantageous in some circumstances, but can be the source of problematic bugs. You will see further instances of the define directive later in the text. It is good convention to write defined words in all capitals, so a programmer will know that this is not a variable that you have declared but a defined macro. It is not necessary to end a preprocessor directive such as define with a semicolon; in fact, some compilers may warn you about unnecessary tokens in your code if you do. Scope[edit] In the Basic Concepts section, the concept of scope was introduced.