Click to prove you're human

Calculating percentages is a fundamental skill in mathematics that we encounter in various aspects of our lives. Whether you're a student tackling homework problems or a professional dealing with financial data, understanding how to compute percentages accurately is essential. Luckily, with the advent of advanced calculators like the TI 84 Plus, performing percentage calculations has become more efficient and convenient. In this step-by-step guide, we will explore how to unleash the power of the TI 84 Plus and effortlessly calculate percentages, saving valuable time and reducing the chances of errors. The TI 84 Plus is a graphing calculator that offers a wide range of functions, making it a popular choice among students, teachers, and professionals alike. By performing percentage calculations on this sleek device, you can streamline your workflow and tackle complex problems effortlessly. This guide aims to walk you through the process, ensuring that you not only understand how to use the TI 84 Plus for percentage calculations but also gain confidence in applying this valuable skill to various scenarios. So let's dive into the world of percentages and discover the secrets of leveraging the power of the TI 84 Plus calculator. Basic Features and Functions A. Brief explanation of the calculator's layout and buttons The TI 84 Plus calculator is a powerful tool for various mathematical calculations, including percentage calculations. Before diving into the specifics of calculating percentages, it's important to have a basic understanding of the calculator's layout and buttons. The calculator features a rectangular shape with a screen at the top and a keypad below it. The keypad consists of various buttons, each serving a specific function. Some of the important buttons to know when using the TI 84 Plus for percentage calculations include: – Numeric Keypad: This section consists of numbers 0-9, allowing you to input values when performing calculations. – Operation Buttons: These buttons include addition (+), subtraction (-), multiplication (×), division (÷), and the equals (=) button for executing calculations. – Navigation Buttons: The calculator features arrow keys for navigating through menus and moving the cursor. – Mode Button: This button is used to access different modes, such as degree/radian mode, to ensure accurate calculations. – Clear Button: This button clears the calculator's screen and resets any previously entered values or calculations. B. Overview of the main functions used for calculating percentages In addition to familiarizing yourself with the layout and buttons of the TI 84 Plus calculator, it's essential to understand the main functions used for calculating percentages. The primary functions for percentage calculations on the TI 84 Plus include: – Entering Values: This function allows you to input the original number and the percentage value you want to calculate. – Converting Percentage to Decimal: Before performing calculations, percentages should be converted to decimals. This function simplifies the conversion process. – Calculating the Percentage Amount: This function determines the percentage of a number by multiplying the original number by the percentage value. – Calculating the Percentage Change: This function calculates the percentage change between two numbers. – Calculating Markup and Discount: Markup refers to the increase in price, while discount represents the decrease. This function helps determine the markup or discount by using percentages. – Solving Percent Problems: This function assists in solving various types of percent problems, including finding the percent increase or decrease, finding the original quantity, or the final quantity. – Using Parentheses: Parentheses are essential for maintaining the order of operations when performing complex percentage calculations. – Rounding and Approximating: This function allows you to round or approximate percentages to a specific number of decimal places. – Utilizing Memory Functions: The calculator's memory functions help store and recall percentage values for repeated calculations. – Troubleshooting Common Errors: This function helps in identifying and resolving common mistakes or errors that may occur during percentage calculations. IEntering Values A. Step-by-step instructions on entering the original number To begin calculating percentages on the TI 84 Plus, you first need to enter the original number into the calculator. Follow these steps to enter the original number: 1. Turn on your TI 84 Plus calculator by pressing the "On" button. 2. Take a look at the calculator's screen and locate the number pad, which consists of digits 0-9 and the decimal point. 3. Use the number pad to input the original number. For example, if the original number is 150, press the keys 1, 5, and 0 in that order. 4. If the original number is a decimal, use the decimal point button to input the appropriate value. For example, if the original number is 0.75, press the keys 0, ., 7, and 5 in that order. B. Describing how to input the percentage value After entering the original number, you will need to input the percentage value for your calculation. Follow these steps to input the percentage value: 1. Locate the multiplication button on the calculator, denoted by "*". 2. Enter the percentage value using the number pad. For example, if you want to calculate 25%, press the keys 2 and 5 in that order. 3. If the percentage value is less than 1, you may need to convert it to a decimal before inputting it into the calculator. For example, to calculate 0.5%, input the keys 0, ., 0, and 5 in that order. It is important to note that when inputting the percentage value, you do not need to include the percentage symbol (%). The calculator automatically recognizes it as a percentage. After entering both the original number and the percentage value, you are ready to perform various percentage calculations on the TI 84 Plus calculator. Converting Percentage to Decimal A. Explanation of why converting to decimal is necessary When performing calculations involving percentages on the TI 84 Plus calculator, it is essential to convert the percentage to decimal form. This conversion allows for more accurate and straightforward calculations. Percentages are typically represented as fractions out of 100, while decimals are expressed as fractions out of 1. Converting percentages to decimals simplifies mathematical operations by eliminating the need to work with fractions. By converting to decimals, calculations involving percentages can be performed using regular arithmetic operations such as addition, subtraction, multiplication, and division. B. Instructions on how to convert a percentage value to a decimal on the calculator To convert a percentage to a decimal on the TI 84 Plus calculator, follow the step-by-step instructions below: 1. Start by inputting the percentage value. For example, if you have 45%, enter 45. 2. Divide the entered percentage value by 100. To do this, press the "/" button on the calculator's keypad, followed by "100". Press the "Enter" key to perform the division. 3. The calculator will display the decimal equivalent of the percentage. For instance, if you entered 45 and divided by 100, the calculator will show 0.45. By converting percentages to decimals, you can perform various calculations more easily. Whether you're calculating tax, discounts, or markup, working with decimals ensures accurate results and simplifies the overall calculation process. By mastering the conversion between percentages and decimals on the TI 84 Plus calculator, you'll be equipped to tackle a wide range of percentage-based problems and have a solid foundation for further mathematical calculations. Practice using this conversion technique to become comfortable with the calculator's functionality and enhance your problem-solving abilities. Calculating the Percentage Amount Calculating the percentage of a number is a common task that can be easily done using the TI 84 Plus calculator. This section will provide a step-by-step guide on how to find the percentage of a number using the calculator's functions. Step 1: Enter the Original Number The first step in calculating the percentage amount is to enter the original number into the calculator. This can be done by simply inputting the number using the calculator's numeric keypad. For example, if you want to find 30% of 200, you would enter 200 into the calculator. Step 2: Input the Percentage Value After entering the original number, the next step is to input the percentage value that you want to calculate. This can be done by entering the percentage directly using the calculator's numeric keypad. It is important to note that percentages should always be entered as decimals. For example, if you want to find 30% of 200, you would enter 0.3 into the calculator. Step 3: Use the Formula Once you have entered both the original number and the percentage value, you can now calculate the percentage amount using the formula (original number * percentage value). This can be done by using the multiplication button (*) on the calculator. For example, if you want to find 30% of 200, you would multiply 200 by 0.3. The result, in this case, would be 60. After performing the calculation, the calculator will display the percentage amount on the screen. By following these three simple steps, you can easily calculate the percentage amount of any number using the TI 84 Plus calculator. Whether you are working on math problems, financial calculations, or any other situation that involves percentages, the calculator can quickly and accurately provide the answers you need. Percentage change is a measure of the difference between two values as a percentage of the original value. It is commonly used in various fields, including finance, economics, and statistics, to analyze and compare data. Being able to calculate percentage change on the TI 84 Plus is important for understanding trends, growth rates, and changes in quantities over time. B. Instructions on How to Calculate Percentage Change on the TI 84 Plus To calculate the percentage change on the TI 84 Plus, follow these step-by-step instructions: Enter the original value into the calculator. Store the original value in a variable using the STO key. Enter the new value into the calculator. Subtract the original value from the new value. Divide the difference by the original value. Multiply the quotient by 100 to obtain the percentage change. For example, let's say the original value is 100 and the new value is 120. Here's how you would calculate the percentage change: Enter 100 into the calculator. Press the STO key followed by a variable (e.g., ALPHA + A) to store the original value. Enter 120 into the calculator. Subtract the stored original value by pressing the ALPHA key followed by the variable (e.g., ALPHA + A) and subtracting it from the new value using the (-) key. Divide the difference by the stored original value by pressing the division (/) key. Multiply the quotient by 100 by pressing the multiplication (x) key followed by 100. The resulting value, 20, represents a 20% increase from the original value. By following these instructions, you can easily calculate the percentage change between any two values using the TI 84 Plus calculator. Whether you're analyzing financial data, tracking changes in quantities, or comparing statistical data, the ability to calculate percentage change accurately will be a valuable tool in your calculations. Calculating Markup and Discount A. Explanation of markup and discount In this section, we will explore the concepts of markup and discount and how to calculate them using percentages on the TI 84 Plus calculator. Markup refers to the increase in price of a product or service, while discount refers to a reduction in price. Understanding how to calculate markup and discount is essential in various fields like retail, finance, and economics. Markup is typically expressed as a percentage over cost, indicating the additional amount added to the cost to determine the selling price. On the other hand, discount is usually expressed as a percentage reduced from the original price, providing customers with savings. B. Step-by-step guide on calculating markup and discount using percentages on the TI 84 Plus To calculate markup on the TI 84 Plus: 1. Enter the original price or cost into the calculator. 2. Multiply the original price by the markup percentage. 3. Add the result from step 2 to the original price to obtain the selling price. For example, if the original price of an item is $50 and the markup percentage is 30%, the calculation would be as follows: ($50) + ($50 * 0.30) = $50 + $15 = $65 To calculate discount on the TI 84 Plus: 1. Enter the original price into the calculator. 2. Multiply the original price by the discount percentage. 3. Subtract the result from step 2 from the original price to obtain the discounted price. For example, if the original price of an item is $100 and the discount percentage is 25%, the calculation would be as follows: ($100) – ($100 * 0.25) = $100 – $25 = $75 By following these step-by-step instructions, you can easily calculate markup and discount using percentages on the TI 84 Plus calculator. These calculations can be useful in real-life scenarios such as determining the selling price of products, calculating savings during sales, or calculating profit margins. Practice using these calculations to gain confidence and proficiency in applying markup and discount percentages. VISolving Percent Problems A. Understanding the Different Types of Percent Problems Percent problems come in various forms, and it is essential to recognize the different types in order to solve them efficiently using the TI 84 Plus calculator. Common types of percent problems include finding the percent of a number, finding the original number if a percent increase or decrease is given, and finding the percent increase or decrease between two numbers. To illustrate, let's consider an example of each type: 1. Finding the Percent of a Number: If you want to find 30% of 150, you need to calculate what percent 30 is of 150. 2. Finding the Original Number: If you know that a number increased by 25% to become 125, you can use the calculator to find the original number. 3. Finding the Percent Increase or Decrease: If a product's price increased from $50 to $60, you can determine the percentage increase using the TI 84 Plus. B. Utilizing the Calculator to Solve Percent Problems Using the TI 84 Plus calculator, solving percent problems becomes much simpler and quicker. The calculator allows you to input the required values and perform the necessary calculations in a few steps. 1. Finding the Percent of a Number: To find the percent of a number, follow the steps mentioned in Section Enter the original number and multiply it by the percentage value you want to find. 2. Finding the Original Number: To find the original number when a percent increase or decrease is given, follow the steps mentioned in Section Divide the given number by 1 plus the percentage increase or decrease. 3. Finding the Percent Increase or Decrease: To find the percent increase or decrease between two numbers, follow the steps mentioned in Section Subtract the original number from the final number, divide that difference by the original number, and multiply by 100. By following these steps and using the appropriate functions on the TI 84 Plus calculator, you can efficiently solve a wide range of percent problems. In conclusion, the TI 84 Plus calculator offers a reliable and efficient tool for solving percent problems. By understanding the different types of percent problems and utilizing the calculator's functions and features outlined in this guide, you will be able to tackle various percent calculations with ease. Remember to practice and familiarize yourself with the calculator to master percent calculations effectively. Using Parentheses A. Explaining the importance of parentheses in percent calculations When performing complex calculations involving percentages on the TI 84 Plus calculator, it is crucial to understand the significance of using parentheses. Parentheses are used to indicate the order in which calculations should be performed and to ensure accurate results. In percent calculations, parentheses allow you to separate different components of the equation and prevent ambiguity. This becomes particularly important when dealing with multiple operations or when calculating percentage change or percent problems. For instance, consider the calculation (20 + 5) * 10%. Without using parentheses, the calculator would first multiply 5 by 10% and then add the result to 20, which would yield an incorrect answer. However, by enclosing 20 + 5 within parentheses, the calculator knows to add the numbers first and then calculate 10% of the sum. Using parentheses clarifies the intended order of operations and ensures that the calculator correctly interprets the equation. B. Providing instructions on how to use parentheses on the TI 84 Plus To utilize parentheses on the TI 84 Plus calculator, follow these step-by-step instructions: 1. Begin by entering the equation or calculation you wish to perform, including the parentheses where appropriate. 2. To open a parenthesis, press the "(" key located near the bottom-left corner of the calculator. 3. After entering the desired calculation within the parentheses, press the ")" key located next to the "(" key to close the parenthesis. 4. Repeat this process for any additional sets of parentheses needed in your calculation. 5. Once the entire equation, including parentheses, is entered, press the "=" key to obtain the result. It's important to note that the TI 84 Plus calculator allows for nested parentheses, where sets of parentheses can be enclosed within other sets. This feature is especially useful when dealing with complex calculations involving multiple operations. By employing parentheses correctly, you can significantly enhance the accuracy and precision of percentage calculations on the TI 84 Plus calculator. With practice and familiarity, you will become proficient in using parentheses to tackle even the most intricate percentage calculations. X. Display Options A. Overview of different display options available on the calculator The TI 84 Plus calculator offers several display options that can be utilized to enhance the user experience when calculating percentages. These display options include different modes, formats, and settings that can be adjusted based on personal preferences. One important display option is the mode selection. The calculator has two main modes: the standard mode and the scientific mode. The standard mode displays numbers in the decimal format, while the scientific mode uses scientific notation. Users can switch between these modes depending on the level of precision required for their calculations. Additionally, the calculator provides different formats for displaying decimals and fractions. The default format is the Auto setting, which automatically chooses the best format based on the calculated number. However, users can also choose to display decimals as eTher fixed decimal places or as a fraction. B. Explaining how to change the display format for percentage calculations To change the display format for percentage calculations on the TI 84 Plus, follow these steps: 1. Press the [MODE] button on the calculator. 2. Use the arrow keys to highlight the desired mode (Standard or Scientific) and press [ENTER] to select it. To change the display format for decimals or fractions, follow these steps: 1. Press the [MODE] button. 2. Use the arrow keys to highlight "Float," "Frac," or "Auto" and press [ENTER] to select the desired format. If "Float" is selected, decimals will be displayed in the decimal format. If "Frac" is selected, decimals will be displayed as fractions. If "Auto" is selected, the calculator will automatically choose the best format based on the calculated number. It is important to note that changing the display format does not affect the actual calculation performed by the calculator. It only changes how the result is displayed on the screen. By utilizing the display options available on the TI 84 Plus calculator, users can customize the appearance of their calculations to suit their needs and preferences. Whether it's selecting a specific mode for precision or choosing a display format for decimal or fractional values, understanding and utilizing these display options can greatly enhance the accuracy and efficiency of percentage calculations on the TI 84 Plus. Rounding and Approximating Importance of rounding and approximating in percentage calculations Rounding and approximating play crucial roles in percentage calculations as they allow for simplified and more manageable results. While it is important to maintain accuracy in mathematical calculations, rounding and approximating percentages can help in practical scenarios where precise decimal values are not necessary or may be too cumbersome to work with. When dealing with real-world data or when presenting percentages in a more simplified manner, rounding and approximating can provide more meaningful and understandable results. For example, if you are calculating a percentage to determine a discount during a sale, it may be more practical to round to the nearest whole number rather than working with several decimal places. Instructions on how to round or approximate percentages on the TI 84 Plus The TI 84 Plus calculator provides convenient functions for rounding and approximating percentages, making it easy to obtain simpler and more practical values. To round or approximate a percentage on the TI 84 Plus, follow these steps: 1. Start by calculating the percentage using the methods described in the previous sections. 2. Once you have the calculated percentage value displayed on the screen, determine the desired level of rounding or approximation. 3. To round the percentage, use the "round" function on the calculator. Press the "math" button, scroll down to find the "round(" function, and enter the percentage value within the parenthesis. Press "enter" to calculate and display the rounded value. 4. To approximate the percentage, you can choose to round it to a specific number of decimal places. For example, to approximate to one decimal place, use the "round(" function but add the desired number of decimal places inside the parenthesis. For instance, entering "round(percentage, 1)" will round the percentage to one decimal place. By following these steps, you can easily round or approximate percentages to your desired level of precision using the TI 84 Plus calculator. Remember to adjust the level of rounding or approximation based on the specific needs of your calculations or the context in which you are presenting the percentages. In conclusion, rounding and approximating provide a practical way to simplify and present percentages in a more understandable format. The TI 84 Plus calculator offers convenient functions to perform rounding and approximation, allowing for easier manipulation of percentage values and more straightforward calculations. By mastering these techniques, you can confidently work with percentages and utilize them effectively in various real-life scenarios. Utilizing Memory Functions A. Explanation of memory functions on the TI 84 Plus In this section, we will explore the memory functions of the TI 84 Plus calculator and how they can be used to enhance percentage calculations. The TI 84 Plus has various storage registers that allow the user to store values for later use, making it easier to perform repeated calculations without re-entering the same values. The calculator has a total of 26 memory registers, labeled A through Z, providing ample storage space for percentage values and intermediate results. These registers can store both numbers and expressions, allowing you to save calculations for later use. B. Demonstrating how to use memory functions for storing and recalling percentage values To store a percentage value in one of the memory registers, follow these steps: 1. After calculating a percentage or obtaining a result, press the "STO¬" button followed by the desired letter (e.g., "STO¬A") to store the value in register A. 2. The stored value will now be saved in memory and can be recalled at any time by pressing the "RCL" button followed by the corresponding letter (e.g., "RCL A"). 3. You can use the recalled value in further calculations or manipulate it using other functions. One advantage of using memory functions is that you can perform calculations involving multiple percentage values. For example, if you need to find the average of two percentages, you can store them in different registers and recall them as needed. Additionally, memory functions can be extremely useful when solving complex percent problems or equations that require the use of multiple percentages. By storing intermediate results in memory registers, you can easily access them at any point during the calculation process. Remember to clear any unwanted values from memory by pressing the "CLEAR" button followed by the desired letter (e.g., "CLEAR A") to ensure accurate results in future calculations. Using memory functions can save time and effort, especially when dealing with repetitive calculations. It allows you to focus on the problem at hand without worrying about retyping values or losing previously obtained results. To become proficient in using memory functions, practice by storing different percentage values and recalling them for various calculations. This will help you familiarize yourself with the process and increase your efficiency when working with percentages on the TI 84 Plus calculator. Conclusion Recap of the step-by-step guide for calculating percentages on the TI 84 Plus In this comprehensive guide, we have provided a step-by-step approach to calculating percentages on the TI 84 Plus calculator. Let's recap the key points covered in the previous sections: First, we started with an overview of the TI 84 Plus calculator and emphasized the importance of knowing how to calculate percentages on this device. Next, we discussed the basic features and functions of the calculator, including its layout and the main functions used for percentage calculations. We then moved on to entering values, providing detailed instructions on inputting the original number and the percentage value. Converting the percentage to a decimal was the focus of the next section. We explained why converting to decimal is necessary and provided instructions on how to do it on the calculator. Afterwards, we showed you how to calculate the percentage amount using the formula (original number * percentage value), providing a step-by-step guide to simplify the process. In the following section, we tackled the concept of percentage change and demonstrated how to calculate it on the TI 84 Plus. Moving forward, we discussed the calculation of markup and discount using percentages on the calculator, providing a step-by-step guide for each. We then explored different types of percent problems and how to solve them using the calculator. To ensure accurate calculations, we emphasized the importance of using parentheses and provided instructions on how to incorporate them into your percentage calculations on the TI 84 Plus. Next, we discussed the display options available on the calculator and how to change the format for percentage calculations. We also highlighted the significance of rounding and approximating in percentage calculations and explained how to do it on the TI 84 Plus. Utilizing memory functions on the TI 84 Plus was the focus of the next section. We explained what they are and demonstrated how to use them for storing and recalling percentage values. Additionally, we addressed common mistakes and errors that may occur when calculating percentages. We provided solutions to overcome these errors specifically on the TI 84 Plus calculator. In conclusion, mastering percentage calculations on the TI 84 Plus can be a valuable skill, particularly when working with data representation or solving real-world problems. By following the step-by-step guide outlined in this article, students, professionals, and anyone else using the TI 84 Plus can become confident in performing accurate percentage calculations. Practice is key, so we encourage you to utilize the calculator's features and functions regularly to improve your proficiency. You can control your preferences for how we use cookies to collect and use information while you're on TI websites by adjusting the status of these categories. Category Description Allow Analytics and performance cookies These cookies, including cookies from Google Analytics, allow us to recognize and count the number of visitors on TI sites and see how visitors navigate our sites. This helps us improve the way TI sites work (for example, by making it easier for you to find information on the site). Advertising and marketing cookies These cookies enable interest-based advertising on TI sites and third-party websites using information you make available to us when you interact with our sites. Interest-based ads are displayed to you based on cookies linked to your online activities, such as viewing products on our sites. We may also share this information with third parties for these purposes. These cookies help us tailor advertisements to better match your interests, manage the frequency with which you see an advertisement, and understand the effectiveness of our advertising. Functional cookies These cookies help identify who you are and store your activity and account information in order to deliver enhanced functionality, including a more personalized and relevant experience on our sites. If you do not allow these cookies, some or all site features and services may not function properly. If you do not allow these cookies, some or all of the site features and services may not function properly. Social media cookies These cookies allow identification of users and content connected to online social media, such as Facebook, Twitter and other social media platforms, and help TI improve its social media outreach. Strictly necessary These cookies are necessary for the operation of TI sites or to fulfill your requests (for example, to track what items you have placed into your cart on the TI.com, to access secure areas of the TI site, or to manage your configured cookie preferences). Always On When you see this sign: + plus add plus add ++++++++ , but not when you see : A+, B+, C+, D+. That is your grade. By: super answerer The % Command The % symbol is an undocumented command on the TI-83 series calculators starting with OS version 1.15. It's useful as a shortcut for percents - it divides by 100, so it will convert numbers to percentages. For example, 50% will become 50/100 or 1/2, which is just what 50% should be. Although this trick can save you a few bytes, it also makes your program incompatible with old OS versions — it's up to you to decide if the tradeoff is worth it. The % symbol is not quite equivalent to the value 0.01: typing in % by itself will give you a syntax error, as expected. Entering the % symbol There are several assembly programs out there that can let you access the % symbol if you know what you're doing, but here is a short, self-contained way. First, create an assembly program by entering the following into the program editor: :AsmPrgmEFF1423605C9 :BBDA Then compile it: for example, if you entered the above into prgmX, and prgmY is free, then you can run Then run the compiled assembly program: Now the compiled assembly program will become unlocked and contain the characters: Most of this is garbage data and can be deleted, and the final character is the % character we wanted. (If you delete the other characters, then the % symbol can be accessed at any time by pressing [2nd][RCL] and choosing prgmY.) Error Conditions ERR:INVALID is thrown on older operating system versions. commandoperatorsperenttokensundocumented Page 2 TI-83 Plus Large Font This fixed-width character set is generally found on the home screen. For the smaller font, please see this article. Character Map The order of the characters is rows then columns, so that is 01 Click on any character to jump ahead and view its respective details. 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 A B C D E F The following chart depicts each character with its hexadecimal equivalent, name, and possible BBCode representation (the latter of which need only be copied and pasted directly into a message board post). Most likely, any numeric character references (e.g., ►) included will be automatically converted upon previewing or sending the message. While there are more accurate Unicode-based substitutes for many of these characters, a rather large user base lacks the fonts or settings necessary to render them in a visually useful manner, or at all. Consequently, precision as there might have been in more well-suited environments has been sacrificed to permit a higher rate of good cross-browser performance. For instances where a sufficient alternative does not exist, image tags with the appropriate URI will be provided. If you wish to use only 7-bit ASCII, then refer to the article on ASCII Output Codes. Determining Font Sizes There is no set standard for BBCode font sizes. Message board vendors will use their own rules of determination, so using the same value for each will produce dissimilar results. Luckily, the representations on this page will only utilize one such transformation, thus lending themselves to the simplified requirement of a single lookup: Whenever you see "[size=x]", the "x" should be replaced by... 0 if using IPB 2.1.7, MyBB, or vBulletin 10 if using phpBB 6pt if using SMF These are based on the default skins for the above vendors. Unsupported Tags If you find that a specific message board doesn't support one or more of the tags provided, then you should use the more accommodating ASCII Output Codes instead. Individual Characters Hex: 01 Name: "Recursive n" BBCode: [font=courier new][i]n[/i][/font] Hex: 02 Name: "Recursive u" BBCode: u Hex: 03 Name: "Recursive v" BBCode: v Hex: 04 Name: "Recursive w" BBCode: w Hex: 05 Name: "Convert" BBCode: [font=courier new]►[/font] Hex: 06 Name: "Square Up" BBCode: [img] /img] Hex: 07 Name: "Square Down" BBCode: [img] /img] Hex: 08 Name: "Integral" BBCode: [font=courier new]∫[/font] Hex: 09 Name: "Cross" BBCode: [font=verdana]×[/font] Hex: 0A Name: "Box Icon" BBCode: [font=courier new]□[/font] Hex: 0B Name: "Cross Icon" BBCode: [font=arial][size=x]+[/size][/font] (?) Hex: 0C Name: "Dot Icon" BBCode: [font=courier new]·[/font] Hex: 0D Name: "Subscript T" BBCode: [font=verdana][size=x]т[/size][/font] (?) Hex: 0E Name: "Cube Root" BBCode: [font=verdana]³[/font] Hex: 0F Name: "Hexadecimal F" BBCode: [font=verdana][b]F[/b][/font] Hex: 10 Name: "Root" BBCode: [font=arial]√[/font] Hex: 11 Name: "Inverse" BBCode: [font=courier new]⁻¹[/font] Hex: 12 Name: "Square" BBCode: [font=verdana]²[/font] Hex: 13 Name: "Angle" BBCode: [img] /img] Hex: 14 Name: "Degree" BBCode: [font=times new roman]°[/font] Hex: 15 Name: "Radian" BBCode: [font=verdana][sup]r[/sup][/font] Hex: 16 Name: "Transpose" BBCode: [font=arial][sup]т[/sup][/font] Hex: 17 Name: "Less Than or Equal To" BBCode: [font=verdana]≤[/font] Hex: 18 Name: "Not Equal To" BBCode: [font=verdana]≠[/font] Hex: 19 Name: "Greater Than or Equal To" BBCode: [font=verdana]≥[/font] Hex: 1A Name: "Negation" BBCode: [font=courier new]¯[/font] Hex: 1B Name: "Exponent" BBCode: [font=verdana][size=x]E[/size][/font] (?) Hex: 1C Name: "Store" BBCode: [font=courier new]→[/font] Hex: 1D Name: "Ten" BBCode: [font=courier new][size=x]10[/size][/font] (?) Hex: 1E Name: "Up Arrow" BBCode: [font=times new roman]↑[/font] Hex: 1F Name: "Down Arrow" BBCode: [font=times new roman]↓[/font] Hex: 20 Name: "Space" BBCode: [font=courier new]▒[/font] Hex: 21 Name: "Exclamation Mark" BBCode: ! Hex: 22 Name: "Quotation Mark" BBCode: [font=verdana]"[/font] Hex: 23 Name: "Pound Sign" BBCode: # Hex: 24 Name: "Fourth" BBCode: [font=verdana][sup]4[/sup][/font] Hex: 25 Name: "Percent Sign" BBCode: % Hex: 26 Name: "Ampersand" BBCode: [font=arial]&[/font] Hex: 27 Name: "Apostrophe" BBCode: ' Hex: 28 Name: "Left Parenthesis" BBCode: (Hex: 29 Name: "Right Parenthesis" BBCode:) Hex: 2A Name: "Asterisk" BBCode: [font=verdana]*[/font] Hex: 2B Name: "Plus Sign" BBCode: + Hex: 2C Name: "Comma" BBCode: [font=times new roman],[/font] Hex: 2D Name: "Dash" BBCode: − Hex: 2E Name: "Period" BBCode: . Hex: 2F Name: "Slash" BBCode: / Hex: 30 Name: "0" BBCode: 0 Hex: 31 Name: "1" BBCode: 1 Hex: 32 Name: "2" BBCode: 2 Hex: 33 Name: "3" BBCode: 3 Hex: 34 Name: "4" BBCode: 4 Hex: 35 Name: "5" BBCode: 5 Hex: 36 Name: "6" BBCode: 6 Hex: 37 Name: "7" BBCode: 7 Hex: 38 Name: "8" BBCode: 8 Hex: 39 Name: "9" BBCode: 9 Hex: 3A Name: "Colon" BBCode: : Hex: 3B Name: "Semicolon" BBCode: [font=times new romain];[/font] Hex: 3C Name: "Less Than" BBCode: < Hex: 3D Name: "Equal To" BBCode: = Hex: 3E Name: "Greater Than" BBCode: > Hex: 3F Name: "Question Mark" BBCode: ? Hex: 40 Name: "At Sign" BBCode: [font=courier new]@[/font] Hex: 41 Name: "Capital A" BBCode: A Hex: 42 Name: "Capital B" BBCode: B Hex: 43 Name: "Capital C" BBCode: C Hex: 44 Name: "Capital D" BBCode: D Hex: 45 Name: "Capital E" BBCode: E Hex: 46 Name: "Capital F" BBCode: F Hex: 47 Name: "Capital G" BBCode: G Hex: 48 Name: "Capital H" BBCode: H Hex: 49 Name: "Capital I" BBCode: I Hex: 4A Name: "Capital J" BBCode: J Hex: 4B Name: "Capital K" BBCode: K Hex: 4C Name: "Capital L" BBCode: L Hex: 4D Name: "Capital M" BBCode: M Hex: 4E Name: "Capital N" BBCode: N Hex: 4F Name: "Capital O" BBCode: O Hex: 50 Name: "Capital P" BBCode: P Hex: 51 Name: "Capital Q" BBCode: Q Hex: 52 Name: "Capital R" BBCode: R Hex: 53 Name: "Capital S" BBCode: S Hex: 54 Name: "Capital T" BBCode: T Hex: 55 Name: "Capital U" BBCode: U Hex: 56 Name: "Capital V" BBCode: V Hex: 57 Name: "Capital W" BBCode: W Hex: 58 Name: "Capital X" BBCode: X Hex: 59 Name: "Capital Y" BBCode: Y Hex: 5A Name: "Capital Z" BBCode: Z Hex: 5B Name: "Theta" BBCode: [font=arial][i]θ[/i][/font] Hex: 5C Name: "Backslash" BBCode: \ Hex: 5D Name: "Right Bracket" BBCode:] Hex: 5E Name: "Caret" BBCode: ^ Hex: 5F Name: "Underscore" BBCode: _ Hex: 60 Name: "Backquote" BBCode: [font=verdana]'[/font] Hex: 61 Name: "Small A" BBCode: a Hex: 62 Name: "Small B" BBCode: b Hex: 63 Name: "Small C" BBCode: c Hex: 64 Name: "Small D" BBCode: d Hex: 65 Name: "Small E" BBCode: e Hex: 66 Name: "Small F" BBCode: f Hex: 67 Name: "Small G" BBCode: g Hex: 68 Name: "Small H" BBCode: h Hex: 69 Name: "Small I" BBCode: i Hex: 6A Name: "Small J" BBCode: j Hex: 6B Name: "Small K" BBCode: k Hex: 6C Name: "Small L" BBCode: l Hex: 6D Name: "Small M" BBCode: m Hex: 6E Name: "Small N" BBCode: n Hex: 6F Name: "Small O" BBCode: o Hex: 70 Name: "Small P" BBCode: p Hex: 71 Name: "Small Q" BBCode: q Hex: 72 Name: "Small R" BBCode: r Hex: 73 Name: "Small S" BBCode: s Hex: 74 Name: "Small T" BBCode: t Hex: 75 Name: "Small U" BBCode: u Hex: 76 Name: "Small V" BBCode: v Hex: 77 Name: "Small W" BBCode: w Hex: 78 Name: "Small X" BBCode: x Hex: 79 Name: "Small Y" BBCode: y Hex: 7A Name: "Small Z" BBCode: z Hex: 7B Name: "Left Brace" BBCode: { Hex: 7C Name: "Vertical Bar" BBCode: | Hex: 7D Name: "Right Brace" BBCode: } Hex: 7E Name: "Tilde" BBCode: [font=arial]~[/font] Hex: 7F Name: "Inverse Equal To" BBCode: [img] /img] Hex: 80 Name: "Subscript 0" BBCode: [font=courier new][size=x]0[/size][/font] (?) Hex: 81 Name: "Subscript 1" BBCode: [font=courier new][size=x]1[/size][/font] Hex: 82 Name: "Subscript 2" BBCode: [font=courier new][size=x]2[/size][/font] Hex: 83 Name: "Subscript 3" BBCode: [font=courier new][size=x]3[/size][/font] Hex: 84 Name: "Subscript 4" BBCode: [font=courier new][size=x]4[/size][/font] Hex: 85 Name: "Subscript 5" BBCode: [font=courier new][size=x]5[/size][/font] Hex: 86 Name: "Subscript 6" BBCode: [font=courier new][size=x]6[/size][/font] Hex: 87 Name: "Subscript 7" BBCode: [font=courier new][size=x]7[/size][/font] Hex: 88 Name: "Subscript 8" BBCode: [font=courier new][size=x]8[/size][/font] Hex: 89 Name: "Subscript 9" BBCode: [font=courier new][size=x]9[/size][/font] Hex: 8A Name: "Capital A Acute" BBCode: Á Hex: 8B Name: "Capital A Grave" BBCode: À Hex: 8C Name: "Capital A Caret" BBCode: Â Hex: 8D Name: "Capital A Diaeresis" BBCode: Ä Hex: 8E Name: "Small A Acute" BBCode: á Hex: 8F Name: "Small A Grave" BBCode: à Hex: 90 Name: "Small A Caret" BBCode: â Hex: 91 Name: "Small A Diaeresis" BBCode: ä Hex: 92 Name: "Capital E Acute" BBCode: É Hex: 93 Name: "Capital E Grave" BBCode: È Hex: 94 Name: "Capital E Caret" BBCode: Ê Hex: 95 Name: "Capital E Diaeresis" BBCode: Ë Hex: 96 Name: "Small E Acute" BBCode: é Hex: 97 Name: "Small E Grave" BBCode: è Hex: 98 Name: "Small E Caret" BBCode: ê Hex: 99 Name: "Small E Diaeresis" BBCode: ë Hex: 9A Name: "Capital I Acute" BBCode: Í Hex: 9B Name: "Capital I Grave" BBCode: Ì Hex: 9C Name: "Capital I Caret" BBCode: Î Hex: 9D Name: "Capital I Diaeresis" BBCode: Ï Hex: 9E Name: "Small I Acute" BBCode: í Hex: 9F Name: "Small I Grave" BBCode: ì Hex: A0 Name: "Small I Caret" BBCode: î Hex: A1 Name: "Small I Diaeresis" BBCode: ï Hex: A2 Name: "Capital O Acute" BBCode: Ó Hex: A3 Name: "Capital O Grave" BBCode: Ò Hex: A4 Name: "Capital O Caret" BBCode: Ô Hex: A5 Name: "Capital O Diaeresis" BBCode: Ö Hex: A6 Name: "Small O Acute" BBCode: ó Hex: A7 Name: "Small O Grave" BBCode: ò Hex: A8 Name: "Small O Caret" BBCode: ô Hex: A9 Name: "Small O Diaeresis" BBCode: ö Hex: AA Name: "Capital U Acute" BBCode: Ú Hex: AB Name: "Capital U Grave" BBCode: Ù Hex: AC Name: "Capital U Caret" BBCode: Û Hex: AD Name: "Capital U Diaeresis" BBCode: Ü Hex: AE Name: "Small U Acute" BBCode: ú Hex: AF Name: "Small U Grave" BBCode: ù Hex: B0 Name: "Small U Caret" BBCode: û Hex: B1 Name: "Small U Diaeresis" BBCode: ü Hex: B2 Name: "Capital C Cedilla" BBCode: Ç Hex: B3 Name: "Small C Cedilla" BBCode: ç Hex: B4 Name: "Capital N Tilde" BBCode: Ñ Hex: B5 Name: "Small N Tilde" BBCode: ñ Hex: B6 Name: "Accent" BBCode: [font=courier new]´[/font] Hex: B7 Name: "Grave" BBCode: [font=courier new]`[/font] Hex: B8 Name: "Diaeresis" BBCode: [font=courier new]¨[/font] Hex: B9 Name: "Inverted Question Mark" BBCode: ¿ Hex: BA Name: "Inverted Exclamation Mark" BBCode: ¡ Hex: BB Name: "Small Alpha" BBCode: [font=courier new]α[/font] Hex: BC Name: "Small Beta" BBCode: [font=verdana]β[/font] Hex: BD Name: "Small Gamma" BBCode: [font=times new roman]γ[/font] Hex: BE Name: "Capital Delta" BBCode: [font=times new roman][size=x]Δ[/size][/font] (?) Hex: BF Name: "Small Delta" BBCode: [font=times new roman]δ[/font] Hex: C0 Name: "Small Epsilon" BBCode: [font=verdana]ε[/font] Hex: C1 Name: "Left Bracket" BBCode: [Hex: C2 Name: "Small Lambda" BBCode: [font=times new roman]λ[/font] Hex: C3 Name: "Small Mu" BBCode: [font=verdana]μ[/font] Hex: C4 Name: "Small Pi" BBCode: [font=times new roman]π[/font] Hex: C5 Name: "Small Rho" BBCode: [font=courier new]ρ[/font] Hex: C6 Name: "Capital Sigma" BBCode: [font=arial]Σ[/font] Hex: C7 Name: "Small Sigma" BBCode: [font=arial]σ[/font] Hex: C8 Name: "Small Tau" BBCode: [font=times new roman]τ[/font] Hex: C9 Name: "Small Phi" BBCode: [font=times new roman]φ[/font] Hex: CA Name: "Capital Omega" BBCode: [font=times new roman]Ω[/font] Hex: CB Name: "X Mean" BBCode: [img] /img] Hex: CC Name: "Y Mean" BBCode: [img] /img] Hex: CD Name: "Superscript X" BBCode: [font=times new roman][sup]×[/sup][/font] Hex: CE

hanging around there. Games Q: Where can I find TI-Basic games and programs to download? A: On our resources page, you will find several links to general TI related sites. One of the best sites to visit for games and programs is ticalc.org, which has the largest archive of TI-Basic games and programs of any site on the Internet. You can also browse our own program archives, which can be found here. Q: What is an emulator? A: An emulator allows you to run a virtual form of your calculator on your computer, which is very convenient when you want to make quick changes to programs, or do any debugging or optimizing. There are several emulators available for you to use, so you should just experiment to see which one you prefer. Q: I downloaded an emulator for my calculator, but it won't work because it says it needs a ROM image. What is that? A: A ROM image is simply an instance of your calculator, which tells the emulator that you own your calculator. It is primarily used as a safeguard because only one person is supposed to be using any one ROM image. To download the ROM image to your computer, you just link your calculator to your computer, and then the emulator should be able to download the ROM image off of it. Q: I have an awesome idea for a game, but I don't know how to program. Can you program it for me? A: While we would like to help you program your game, we each have our own projects that we're working on and other real-world things (like school and a job) that occupy our time, so we aren't able to program your game for you. At the same time, if you have a specific TI-Basic programming question that you need help with, we'd be happy to help you. Even better than us programming your game, though, is you programming it yourself (see next question). Q: What do I need to make games? A: The main things you need to make games are your TI calculator and calculator manual. Before you actually implement a game, however, you should plan it out. This involves coming up with the idea for the game and working out the many details of the game: graphics, gameplay, menus, and so on. Once you have all of those things figured out, you just need to put them into action. Q: What is a good tutorial for making games? A: Unfortunately, there really is no comprehensive game tutorial available. Instead, there are several small tutorials that each cover different aspects of games. In addition, on this wiki there are quite a few techniques covered — see for example the Special Topics section. Q: Can I use a routine from this wiki in my game? A: Yes! In fact, we encourage it. All of the routines on this site are designed to be as optimized and efficient as possible, so that readers learn the best way to program. Q: Can I use sprites from other games in my own game? A: The general consensus among the calculator programming community is that using somebody else's graphics in your game is fine, as long as you get their permission to do so. However, if you don't plan on releasing your game to the community, but instead just keeping it to yourself and your friends, then it doesn't really matter. Programming Q: How do I draw graphics? A: 68k TI-Basic has many graphics commands. Probably the most useful ones are the *Pic commands (such as XorPic, which can be used to display images of any size anywhere on the screen. However, other commands, such as Line are also useful. Q: Can I do [task] in TI-Basic? A: While it's possible to do almost anything in TI-Basic, whether it looks nice and runs at a decent speed is a different matter. If you have thoroughly planned your program and made it as optimized as possible, and your program still takes a minute to load and there's a five second lag after each key press, that's a good indicator that you should probably use assembly or C instead. At the same time, you should always strive to push the boundaries of TI-Basic. Q: How do I convert a number to a string and vice versa? A: The string() command can be used to convert any variable type to a string. To convert in the other direction, use the expr() command. Q: My program is extremely large. Is there a way to manage/condense the code better? A: First of all, your program will probably shrink in size after the first time you run it, due to tokenization. If that's not enough for you, see the optimization page for more tricks. Q: Are there any undocumented features (Easter eggs) in TI-Basic? A: Unlike the TI-83 version of TI-Basic, virtually everything we know about the 68k calculators' TI-Basic is documented somewhere. However, there are some features that the manual doesn't emphasize but are quite useful to TI-Basic programmers. An example is the alternate parameters of the setMode() command. Q: How do you disable the ON key? A: This is impossible in pure TI-Basic. Using Try..EndTry blocks, you can disable the ON key during text input, but the only way to disable it universally is with an assembly program or Exec code. Q: How do I hide the code of my TI-Basic program? A: This is impossible — if someone can run your program, they can see the code as well. You really shouldn't try to hide the code too, but let others learn from it instead. Troubleshooting Q: My calculator can't handle expressions with several variables, (e.g. it can't factor x^2+y^2+2xy). How do I fix this? A: When you type xy next to each other, the calculator doesn't treat it as multiplication, but as a different variable called "xy". Write x*y instead and you should have no trouble. Q: I've entered a simple expression and am confident of the result, but the calculator gives something bizarre! (For example, d(x^2,x) gives 10 as a result) What's wrong? A: Make sure all the variables you're manipulating in a general way (x, in the example) are actually undefined. If not, delete them using DelVar. Otherwise, their values will get substituted and you'll get a weird answer. This can result in other surprising errors as well. Q: I think some of the routines on this wiki have errors in them because they didn't work for me. Could you please correct them? A: We have strived to make sure that all of the routines on this site work correctly and without problems. However, if you are 100% sure that you entered the routine correctly into your calculator, please leave a comment on the page using the comment function at the bottom of the page. Somebody will then be able to correct the routine so that it won't cause anybody else any problems. Q: I was playing a TI-Basic game and my calculator suddenly shut off. When I turned it back on, my memory was erased. What happened? A: Your game had a glitch of some kind, and it caused the calculator to crash. This is usually caused by Assembly programs, as the majority of TI-Basic errors are caught by the calculator. You don't have to worry very much about TI-Basic crashes because they don't do any real permanent damage to the calculator, but because it is very annoying to have to replace all of your programs after your RAM is cleared, you should always store any important files in the archive. Q: When I tried to run my TI-Basic program, I got this error message. What does it mean? A: Most error messages are fairly self-explanatory, but if you're still confused, you should consult our list of error messages with a more in-depth explanation. Q: The transferring program, TI Connect, does not work for me. How do I fix it? A: TI Connectivity issues can be a problem on both the Mac and Windows platforms. Here are some listed solutions for the Windows and Mac platforms. Mac: Uninstall and reinstall TI Connect Kill the process "TI Connect Manager X". This can be done using Activity Monitor. Start Activity Monitor in /Applications/Utilities, then find "TI Connect Manager X" in the list of processes (if you can't find it, just type "ti" on your keyboard). Select it, then click the X icon in the top left corner of Activity Monitor. Click Force Quit. Then you can restart the device manager, and it should detect your calculator. In some cases, disconnecting your calculator or turning it off will always cause TI Connect Manager X to crash, and you will need to kill it every time you need to connect it to your computer. To make this process easier, there is a script here that you can use to automate this process. Use a different USB port. Windows: Reinstallation should fix most problems. Check to make sure the plug (on both ends) are firmly in the port. Assembly/C Q: How does TI-Basic compare to Assembly or C? A: TI-Basic is much easier to learn and program in, but it is rather slow because it is an interpreted language. TI-Basic has many good graphics commands, but will still be slower than assembly or C programs; also, TI-Basic programs are limited in control over the calculator. Q: Is it possible to convert TI-Basic to Assembly? A: No, it is not. There are currently no working programs available that will convert TI-Basic to Assembly (note: I say working because people have tried creating TI-Basic to Assembly converters, but nobody has completed one yet), so the only way you can convert a TI-Basic program to Assembly is by learning Assembly and porting the program yourself. You could also try asking an Assembly programmer to port it for you, but most people won't do that unless the program is pretty small. However, if you are looking to speed up your TI-Basic program, you have the option of using BasicBuilder 3.0, which packages your TI-Basic program into an application. Q: I want to use an Assembly program with my TI-Basic program, but I can't figure out how to use it. Can you help me? A: Unfortunately, we really can't do much for you. What we recommend is that you contact the author of the Assembly program and ask them for help. They wrote the program, so naturally, they should be able to answer any questions that you have. [[/div]] Page 9 Click here to edit contents of this page. Click here to toggle editing of individual sections of the page (if possible). Watch headings for an "edit" link when available. Append content without editing the whole page source. Check out how this page has evolved in the past. If you want to discuss contents of this page - this is the easiest way to do it. View and manage file attachments for this page. A few useful tools to manage this Site. See pages that link to and include this page. Change the name (also URL address, possibly the category) of the page. View wiki source for this page without editing. View/set parent page (used for creating breadcrumbs and structured layout). Notify administrators if there is objectionable content in this page. Something does not work as expected? Find out what you can do. General Wikidot.com documentation and help section. Wikidot.com Terms of Service - what you can, what you should not etc. Wikidot.com Privacy Policy. Page 10 Strings and Their Commands A string is a collection of letters (usually called characters) in order. They are most commonly used for displaying text, although they can be adapted to any purpose. See the character codes page for a table of the characters that can be part of a string. To write a string, use quote marks around the characters you want it to contain. For example, "Hello" is a string containing the characters H, e, l, l, and o, in that order. Using the string() command, you can also create a string out of any other variable type. On the TI-68k calculators, strings have an advantage over lists or matrices - they are a random access structure, which means that accessing the end of the string is just as fast as accessing the end of the string (for a list, on the other hand, the calculator has to go through every previous element of the list to get to an element at the end). Although strings are awkward to access normally, so short strings are at a disadvantage compared to lists, long enough strings will beat out lists and matrices, at least for accessing a specific character. Except for the constraint of memory, there is no limit to the amount of characters in a string. There are two special operators that can be used on strings - & (concatenation) and # (indirection). The concatenation operator, &, joins two strings together - "Hello"&" world" returns "Hello world". The indirection operator, #, replaces a string containing the name of a variable by the variable itself - #"f"(x) is the same as f(x) (this can be very powerful, especially for non-algebraic variables such as pictures). (annoyingly, there is no easy way to type #, on a TI-89. You can select it from the character menu (2nd CHAR, 3, 3) or from the catalog) The relational operators (=, ≠, >, ≥, Page 14 Lists and Their Commands A list is a collection of elements in order. On the TI-68k calculators, lists can contain any mix of any variable type that's valid in an expression: you can have lists of numbers, lists of strings, lists of truth values or expressions. You can even mix and match variable types — it's perfectly all right to have a string in one element, and a number in the next. The only special case is lists of lists: these are kind of allowed, but they're called matrices, and have some further restrictions. Lists are written using curly brackets { and } around the elements, separated by commas. For example, {1,2,3,4,5} is a list containing the numbers 1 through 5 in that order. You can access a certain element of the list by writing the number of the element you want in [ ] brackets after it: listvar[5] would select the 5th element of listvar. Elements are numbered starting with 1. On earlier calculator models, lists had the random access property: accessing any element of a list took the same amount of time. This was possible because the lists were restricted to numbers. On the 68k calculators, since lists can mix element types, they are no longer random access: the calculator has to go through the entire list to get to an element, so the larger an index is, the longer it takes to access. This isn't significant for short lists. But taking the 100th element, for example, takes approximately twice as long as taking the 1st element, and the time keeps increasing linearly, so it can be very slow to access the last elements of a long list. Except for the constraint of free memory, and of the time it takes to access elements, there is no limit on the number of elements a list may have. Operations on Lists Many commands, including math commands and others, can be extended to lists by applying them to each element of the list. An example is sin(): sin({1,2,3}) {sin(1) sin(2) sin(3)} If a command has more than one argument, there are two ways to extend it to apply to lists. One is to use it with a list and a regular argument: then the command will be applied to each element of the list paired up with the regular argument. Here is this way illustrated with mod() mod({10,20,30},7) {3 6 2} The other way is to make both arguments lists. In that case, the lists must be the same size, and each element of the first list will be paired with the corresponding element of the second. For example: mod({10,20,30}, {7,8,9}) {3 4 3} Although in these examples, mod() could be extended in both ways, sometimes only one is possible. PtOn, for example, (as well as other point commands) can be used with two numbers, or two lists, but not with a list and a number. round(), on the other hand, can be used with two numbers, or a list and a number, but will give a meaningless expression when applied to two lists. A noteworthy special case is the basic math operators (+, -, *, /, ¯, and ^), which can all be used with lists (in both ways). List-Specific Commands Of course, there are commands specifically designed for use with lists. Several of these, such as dim() or rotate(), can be also used with strings. Many of these commands are found in the list menu (Press 2nd MATH to access the popup math menu, then select 3:List). augment() crossP() cumSum() dim() dotP() exp▶list() Fill left() Δlist() list▶mat() mat▶list() max() mid() min() newList() polyEval() product() right() rotate() seq() shift() SortA SortD sum() Several statistics commands can be applied to lists as well. Conditionals With Lists Conditional statements, like If, when(), and While, accept lists of truth values as well as single truth values. The check will be interpreted as true if and only if every element of the list is true, effectively combining each element of the list with and. The most common way for lists of truth values to be created is with relational operators (=, ≠, >, ≥, , Page 17 Click here to edit contents of this page. Click here to toggle editing of individual sections of the page (if possible). Watch headings for an "edit" link when available. Append content without editing the whole page source. Check out how this page has evolved in the past. If you want to discuss contents of this page - this is the easiest way to do it. View and manage file attachments for this page. A few useful tools to manage this Site. See pages that link to and include this page. Change the name (also URL address, possibly the category) of the page. View wiki source for this page without editing. View/set parent page (used for creating breadcrumbs and structured layout). Notify administrators if there is objectionable content in this page. Something does not work as expected? Find out what you can do. General Wikidot.com documentation and help section. Wikidot.com Terms of Service - what you can, what you should not etc. Wikidot.com Privacy Policy. Page 18 Errors This table lists all the possible error messages that can occur. The error number to the left is not displayed, but is stored to the errornum system variable, which can be used in a Try..EndTry block. Some error messages are very explicit (like # 810), but others are less so. Where the error message itself gave too little information, this table explains the error more thoroughly. A large part of the information here was taken from Appendix B of the TI-89 manual. Error Number Error Description 10 A function did not return a value 20 A test did not resolve to TRUE or FALSE This error usually occurs when comparing an undefined variable, in a statement such as If. 30 Argument cannot be a folder name 40 Argument error 50 Argument mismatch Two of the arguments must be of the same type. For example, PtOn can be used with two numbers, or two lists. But a number and a list can't be used together. 60 Argument must be a Boolean expression or integer For use with logic commands such as or They can be applied to two truth values, or bitwise to two integers. 70 Argument must be a decimal number 80 Argument must be a label name 90 Argument must be a list 100 Argument must be a matrix 110 Argument must be a Pic 120 Argument must be a Pic or string Happens with a Title used in a toolbar. The "Pic" part only applies to the widescreen calculators, not the 89 or 89 Titanium 130 Argument must be a string 140 Argument must be a variable name This can also indicate an invalid variable name such as 1xy. 150 Argument must be an empty folder name Folders can't be deleted if they're not empty. 160 Argument must be an expression For example, zeros(2x+3=0,x) is not valid because 2x+3=0 is an equation. Use zeros(2x+3,x) instead — the =0 is implied with this command. 161 ASAP or Exec string too long This error is caused by the assembly program RAM limit. The limit is not present on HW1 calculators; on others, it's 8k with AMS 2.03 or lower, and 24k with AMS 2.04 or higher. If you try to run an assembly program that exceeds the limit, you'll get this error. 163 Attribute (8-digit number) of object (8-digit number) not found 165 Batteries too low for sending/receiving product code 170 Bound The lower bound must be less than the upper bound. This error can happen, for example, with the zero finder on the graph. 180 Break The ON key was pressed during a calculation or while running a program. This error usually can't be caught by Try..EndTry blocks, unless it happens in a text prompt. 185 Checksum error 190 Circular definition Circular definitions of a variable are caught: for example, a+1→a (if a is undefined). But circular definitions of a function are handled by a limit on recursion depth. 200 Constraint expression invalid See the page for the | (with) operator for more details. 210 Data type An argument is of the wrong data type. 220 Dependent limit The same variable can't be used as both an integration variable and a bound. For example, ∫(sin(x),x,0,x) wouldn't be allowed. 225 Diff Eq setup 230 Dimension An index went out of the bounds of a list or matrix. For example, list[5] when list is equal to {1,2,3,4}. 240 Dimension mismatch Some commands require their list or matrix arguments to match in size. For example, you can't add the lists {1,2} and {1,2,3}. 250 Divide by zero Take that, James Anderson! 260 Domain error Some commands only accept numbers within a certain range. For example, ans() only works with numbers 1-99. 270 Duplicate variable name 280 Else and ElseIf invalid outside If..Then block Well, actually, Try..EndTry blocks also use Else. But you didn't hear me say that. 290 EndTry is missing the matching Else statement 295 Excessive iteration Happens when an iterative solver runs for too long without finding a solution. This usually means there isn't one, except in truly horrible cases. 300 Expected 2 or 3-element list or matrix This happens with commands that deal with 2 or 3-dimensional vectors. 307 Flash application extension (function or program) not found 308 Flash application not found 310 First argument of nSolve must be a univariate expression English translation: the only undefined variable that can be in nSolve()'s expression is the one you're solving for. 320 First argument of solve or cSolve must be an equation or inequality For example, solve(2x+3,x) is invalid because 2x+3 isn't an equation. Of course, zeros() has exactly the opposite problem. 330 Folder Sometimes the calculator likes being laconic. This error happens in the VAR-LINK menu if a variable is stored to a folder that doesn't exist. 335 Graph functions y1(x)..y99(x) not available in Diff Equations mode 345 Inconsistent units 350 Index out of range 360 Indirection string is not a valid variable name 380 Invalid ans() That is, that many answers haven't been stored yet. 390 Invalid assignment 400 Invalid assignment value 405 Invalid axes 410 Invalid command 420 Invalid folder name 430 Invalid for the current mode settings 440 Invalid implied multiply The syntax a(b) is only used for function calls, not for multiplying a and b. If a is not a defined function, the calculator assumes you tried to multiply, and gives this error. 450 Invalid in a function or current expression A user-defined function can't change global variables, or use certain commands. This also happens in prompts, such as the Data/Matrix Editor. 460 Invalid in Custom..EndCustm block 470 Invalid in Dialog..EndDlog block 480 Invalid in Toolbar..EndTBar block 490 Invalid in Try..EndTry block 500 Invalid label Label names have the same limitations as variable names. 510 Invalid list or matrix Lists can only be 1-dimensional (lists) or 2-dimensional (lists of lists). Lists of lists are matrices, and any non-rectangular stuff (e.g. {1,{2,3}}) is not allowed. 520 Invalid outside Custom..EndCustm or Toolbar..EndTBar blocks 530 Invalid outside Dialog..EndDlog, Custom..EndCustom, or Toolbar..EndTBar blocks 540 Invalid outside Dialog..EndDlog block 550 Invalid outside function or program 560 Invalid outside Loop..EndLoop, For..EndFor, or While..EndWhile blocks 570 Invalid pathname 580 Invalid polar complex You might think that this is a prohibition against building military bases on the North Pole. Actually, this is used with the ∠ command. For example, (i∠2) is invalid. 590 Invalid syntax block For the miscellaneous errors with Dialog..EndDlog, Custom..EndCustm, and Toolbar..EndTBar blocks. 600 Invalid table 605 Invalid use of units 610 Invalid variable name in a Local statement Reserved variables, for example, can't be made local. 620 Invalid variable or function name The variable name is being used for a built-in function. 630 Invalid variable reference 640 Invalid vector syntax 650 Link transmission 665 Matrix not diagonalizable 670 or 673 Memory 680 Missing (690 Missing) 700 Missing " 710 Missing] 720 Missing } 730 Missing start or end of block syntax For example, an If without an EndIf to go with it. 740 Missing Then in the If..EndIf block 750 Name is not a function or program 765 No functions selected 780 No solution found Most commands usually think of something clever to return, instead. So this error only occurs with the interactive graph tools. 790 Non-algebraic variable in expression 800 Non-real result Only happens if the calculator is in real number mode. 810 Not enough memory to save current variable. Please delete unneeded variables on the Var-Link screen and re-open editor as current OR re-open editor and use F1 8 to clear editor. 830 Overflow The possible range of a floating point number is between -101000 and 101000 (not inclusive). However, sometimes an overflow is replaced by infinity instead. 840 Plot setup 850 Program not found 860 Recursion is limited to 255 calls deep 870 Reserved name or system variable System variables cannot be deleted, for example. 875 ROM-resident routine not available 880 Sequence setup 885 Signature error 890 Singular matrix 895 Slope fields need one selected function and are used for 1st-order equations only 900 Stat Statistics. 910 Syntax An expression or entry just doesn't make sense. For example, 2+2+. 930 Too few arguments 940 Too many arguments 950 Too many subscripts "Too many" is more than two. 955 Too many undefined variables 960 Undefined variable 965 Unlicensed product software or Flash application 970 Variable or Flash application in use 980 Variable is locked, protected, or archived 990 Variable name is limited to 8 characters 1000 Window variables domain 1010 Zoom 1020 Internal Error 1030 Protected memory violation Page 19 68k TI-Basic for 83 TI-Basic Programmers This tutorial is meant as an introduction to 68k TI-Basic for programmers that already are fairly experienced with TI-83 series Basic programming. Instead of re-teaching many things, this tutorial highlights the differences between the two languages. Major Features A major novelty of the 68k calculators is the ability to make symbolic calculations. This has many applications: you can deal with expressions such as x2+2*x+1 but treat x as an unknown, or deal with the exact value of √π/3 without approximating it with floating-point values. The calculator no longer has some statistical commands, but has much more powerful calculus commands (it can do symbolic derivatives, integrals, and finite and infinite sums, among other things). This doesn't have any immediate programming applications, although you can often find an unexpected use for these commands. A very programming-relevant difference, on the other hand, is the advent of error catching. Using the Try and EndTry statements, your program can identify if an error has occurred, and possibly even recover from this (or at least display a custom error message). Also one of the highlights is how much pictures have been empowered. They now can be any size and be displayed with any logic. True "real-time" multiplayer games is now possible with the SendCalc Command, something that was impossible on 83's. The more specific differences described below tend to combine to make programs run faster, and allow for a programming style closer to programming a more "serious" language on a computer. Commands On the TI-89, commands can be entered letter by letter, and don't have to be chosen from a menu. In practice, programs and functions are tokenized, making a command range take up 1 to 3 bytes in a program. Many commands have been added or removed between the two languages (see the command index for a full list of commands). In addition, the following commands have changed in spelling: There are two more overall changes. First, many commands' names have been truncated where they were longer than 8 characters: this is the maximum for a command name on the TI-89. An example is RclPic, the 68k equivalent of RecallPic. Second, the use of parentheses after a command now follows a strict convention. "Instructions" — commands that do not return a value — do not require parentheses (e.g. If, Text, etc.) "Functions" — commands that do return a value — require parentheses (e.g. sin(), setMode(), etc.) Even functions with no arguments use parentheses (e.g. getKey(), startTmr(), etc.) Many commands have been added. However, as far as statistics goes, the 68k calculators are inferior, even, to the TI-83 series; most of the functionality is now restricted to regressions, and the calculator doesn't even know internally how to calculate most probability distributions. Variables The way variables are stored has undergone major changes from the TI-83 series. All variables now share a common naming system: the name of a variable can be up to 8 letters long. Variables can also be placed in different folders, which can't be nested but otherwise are very similar to file folders on a computer. By default, variables are stored in the folder 'main'. On the surface, the variable types are much the same as they were on the TI-83 calculators: you have numerical variables, lists, matrices, strings, picture variables, equations, graph databases, and a new one, called "data". These are slightly different, however. To begin with, numerical variables can now be either floating-point (the same as on the TI-83 series) or integer variables (that don't have a decimal place, but have several hundred digits of precision). As a consequence of the symbolic operations on a 68k calculator, you also have expressions: formulas that are only evaluated as much as possible. List are very different. They now can hold any combination of numbers, expressions, and character strings. This makes them more powerful, but also slows them down significantly. Data's are basically matrices, but with the new capabilities (and speed limitations) of the new lists. Matrices stayed the same though, limited to numbers, but retained their speed. You can effectively emulate an "old style" list by using a matrix with only 1 row/column, Boolean variables will also cause some confusion for TI-83 programmers: Instead of boolean (true/false) operations returning a one or a zero that can be used in the same manner as regular integers (1, 3, -5, 4.8, etc.), boolean variables now all have a class of their own. This means that you can't use a simple "If (variable)" statement to check whether a variable has yet been defined, and can't use shortcuts in expressions that involve a parenthetical boolean statement determining whether a certain term in the expression is used (e.x. (k>2)(x+1)+3x). Instead, you will have to use the ifVar() command to check whether a variable has been defined and return to regular If-Then blocks for conditional computing. Programs Programs are also considered variables, on the same level as any other: you can even define a program within another program. They imitate built-in commands, and can even be given parameters. Using the Local command, you can declare local variables that are reset to their old values once a program finishes running. You can also define functions, which are similar to programs but return a value. Functions have some other limitations, though: they can only use local variables, and can't modify any global aspects of the calculator (so graphical commands, for example, are limited to programs). With local variables, and the ability to define functions and programs, you can program in a procedural language style. Instead of placing the entire code of the program in one block, you can split it up into functions and subprograms that are defined at the beginning of the main program. The entire issue of memory leaks (caused on the TI-83 series by jumping out of code blocks with Goto) is no longer present in 68k TI-Basic. Loops have offsets linking the end to the beginning, so the program doesn't need to keep a stack to be aware of what to do with End instructions. There is no longer any memory cost to entering a loop (or any other kind of code block), so it's impossible to leak memory this way. Optimizations Most types of trivial optimizations from the TI-83 series are invalid on the 68k calculators. For example, closing parentheses, quotes, and brackets are now mandatory — but don't add any size to the program, since it's tokenized and converted to postfix notation. The Ans variable no longer plays an important role: though the ans() command does exist to replace it, it's not modified by storing to variables inside a program, so it's mostly useful on the home screen. A large part of 68k optimization revolves around careful use of lists. List variables are no longer random access: accessing the last element of a list is much slower than accessing the first element. For this reason, going through a list in a For loop is about the worst thing you could do. Graphics 68k gives the programmer much more options with displaying graphics. Sprites, which had to be displayed using various tricks or libraries on a TI-83 series calculator, are built into 68k as picture variables. These picture variables bear little resemblance to the screenshot-like functionality they have on a TI-83+. They can be any size from 1x1 to the entire screen, and can be stored from and recalled to any part of the graph screen. There are even several commands for displaying a sprite using different logic. Apart from these very powerful commands, more ordinary commands have also been buffed up. Virtually all graphics commands have a point and a pixel equivalent, so you're free to choose one or the other to use (usually, you'll want pixels). The Circle command now draws circles instantaneously, as opposed to taking several seconds. Instead of being forced to choose between home screen and graph screen, the choice is between graph screen and "Program I/O" screen on the 68k calculators. The program I/O screen is a separate home screen for programs, which is limited to text (but the text doesn't have to be aligned). In addition, both screens can be spiced up using dialogs, which imitate the appearance of a popup window on a computer, and are great for inputting data without having to erase anything from the screen. Another major addition to the graphics command set is the newly created Dialog feature. New commands add extra I/O capability that doesn't interfere with the program I/O or the graph screen, allowing for enhanced in-program data entry. There is one limit to graphics on the 68k calculators - they cannot draw over the top menu bar and bottom status bar, so are effectively limited to only 2/3 of the screen. Assembly libraries can be used to access the entire screen, but this is impossible in TI-Basic alone. Closing Words This page gives an overview of some of the features of 68k, but it isn't, and cannot, be complete. There are other pages you could visit to get a better picture of 68k programming: Command Index FAQ Sample Programs However, the best way to try to learn the language is first-hand experience with it. Page 20 Key Codes This table contains the values returned by getKey() for each (keypress, modifier) pair. Unlike the getKey found on TI-83 series calculators, the 68k calculators' getKey() returns keypresses with modifiers. On the TI-92, TI-92+, and Voyage 200, the key layout is different. The same effect (e.g. F8) still corresponds to the same value (e.g. 275), but is obtained differently (by pressing F8, in this example, rather than 2nd+F3). The alpha key is replaced by the "grab" button for the arrow keys. Notice that the key codes for typing a character correspond to that characters character code. For example, the key code for typing "A" is 65, and ord("A") will also return 65. This even applies to the key codes for the international characters that aren't shown in this table. Key Modifier None ⇧ (shift) 2nd ♦ (diamond) alpha Result Value Result Value Result Value Result Value Result Value Result Value F1 F1 268 F1 268 F6 273 Y= 8460 F1 268 F2 F2 269 F2 269 F7 274 WINDOW 8461 F2 269 F3 F3 270 F3 270 F8 275 GRAPH 8462 F3 270 F4 F4 271 F4 271 F4 271 TblSet 8463 F4 271 F5 F5 272 F5 272 F5 272 TABLE 8464 F5 272 ♦ (diamond) copy 24576 cut 12288 alpha a-lock ESC ESC 264 ESC 264 QUIT 4360 PASTE 8456 ESC 264 APPS APPS 265 APPS 265 SWITCH 4361 8457 APPS 265 ON ON 267 OFF OFF HOME HOME 277 HOME 277 CUST 4373 HOME 277 HOME 277 MODE MODE 266 MODE 266 ▶ 18 _ 95 MODE 266 CATALOG CATLG 278 CATLG 278 i 151 ∞ 190 CATLG 278 ← (backspace) ← 257 ← 257 INS 4353 DEL 8449 ← 257 CLEAR CLEAR 263 CLEAR 263 CLEAR 263 8455 CLEAR 263 X x 120 X 88 ln(4184 e^(8280 x 120 Y y 121 Y 89 sin(4185 sin⁻¹(8281 y 121 Z z 122 Z 90 cos(4186 cos⁻¹(8282 z 122 T t 116 T 84 tan(4180 tan⁻¹(8276 t 116 ^ ^ 94 ^ 94 π 140 θ 136 ^ 94 | | (with) 124 F 70 ° 176 FORMAT 8316 f 102 ((40 B 66 { 123 b 98)) 41 C 67 } 125 © 169 c 99 , , 44 D 68 [91 8236 d 100 ÷ / 47 E 69] 93 ! 33 e 101 × * 42 J 74 √(4138 & 38 j 106 - - 45 O 79 VAR-LINK 4141 contrast + o 111 + + 43 U 85 CHAR 4139 contrast - u 117 ENTER ENTER 13 ENTER 13 ENTRY 4109 APPROX 8205 ENTER 13 STO▶ → 258 P 80 RCL 4354 @ 64 p 112 = = 61 A 65 ' 39 ≠ 157 a 97 EE E 149 K 75 ∠ 159 SYMB 8341 k 107 (-) - 173 SPACE 32 ans(1) 4372 8365 SPACE 32 . . 46 W 87 > 62 ≥ 158 w 119 0 0 48 V 86 < 60 ≤ 156 v 118 1 1 49 Q 81 " 34 8241 q 113 2 2 50 R 82 \ 92 8242 r 114 3 3 51 S 83 UNITS 4147 8243 s 115 4 4 52 L 76 : 58 8244 l 108 5 5 53 M 77 MATH 4149 8245 m 109 6 6 54 N 78 MEM 4150 8246 n 110 7 7 55 G 71 ∫(4151 8247 g 103 8 8 56 H 72 d(4152 8248 h 104 9 9 57 I 73 ; 59 8249 i 105 Arrow key Modifier None ⇧ (shift) 2nd ♦ (diamond) alpha/@ ═ (left) 337 16721 4433 8529 33105 ⇑ (up) 338 16722 4434 8530 33106 ═ (right) 340 16724 4436 8532 33108 ⇓ (down) 344 16728 4440 8536 33112 ⇖ (left+up) 339 16723 4435 8531 33107 ⇗ (right+up) 342 16726 4438 8534 33110 ⇙ (left+down) 345 16729 4441 8537 33113 ⇘ (right+down) 348 16732 4444 8540 33116 An easy way to find the value of a keypress without having to consult this page is to write a short program to output key codes: :Prgm :Local k :0→k :While k=0 :getKey()→k :EndWhile :Text string(k) :EndPrgm Page 21 Character Codes This page contains the character codes used internally by the calculator. This table can be useful in particular with the char() and ord() commands. Characters in the following ranges can be used for variable names: 48...57 (numbers, as long as they don't begin a variable name) 65...90, 97...122 (the usual alphabet, case insensitive) 128...139, 141...148, 181 (Greek letters, case sensitive: ω is not the same as Ω) 192...214, 216...246, 248...255 (international characters, case insensitive) Note that value 2 (STX) cannot be used on the home screen, because for some reason it does not exist in large font. Value Char Value Char Value Char Value Char Value Char Value Char Value Char 0. NUL 32. space 64. @ 96. ‘ 128. α 160. ... 192. À 224. à 1. SOH 33. ! 65. A 97. a 129. β 161. ¡ 193. Á 225. á 2. STX 34. " 66. B 98. b 130. Γ 162. ¢ 194. Â 226. â 3. ETQ 35. # 67. C 99. c 131. γ 163. £ 195. Ã 227. ã 4. EOT 36. $ 68. D 100. d 132. Δ 164. ¤ 196. Ä 228. ä 5. ENQ 37. % 69. E 101. e 133. δ 165. ¥ 197. Å 229. å 6. ACK 38. & 70. F 102. f 134. ε 166. ¦ 198. Æ 230. æ 7. BEL 39. ’ 71. G 103. g 135. ζ 167. § 199. Ç 231. ç 8. BS 40. (72. H 104. h 136. θ 168. √ 200. È 232. è 9. TAB 41.) 73. I 105. i 137. λ 169. © 201. É 233. é 10. LF 42. * 74. J 106. j 138. ξ 170. a 202. Ê 234. ê 11. VT 43. + 75. K 107. k 139. ∏ 171. « 203. Ë 235. ë 12. FF 44. , 76. L 108. l 140. π 172. ¬ 204. Ì 236. ì 13. CR 45. - 77. M 109. m 141. ρ 173. ¯ 205. Í 237. í 14. (lock) 46. . 78. N 110. n 142. ∑ 174. ® 206. Î 238. î 15. ✓ 47. / 79. O 111. o 143. σ 175. - 207. Ï 239. ï 16. ■ 48. 0 80. P 112. p 144. τ 176. ° 208. Ð 240. ð 17. ◀ 49. 1 81. Q 113. q 145. φ 177. ± 209. Ñ 241. ñ 18. ▶ 50. 2 82. R 114. r 146. ψ 178. ² 210. Ò 242. ò 19. ▲ 51. 3 83. S 115. s 147. Ω 179. ³ 211. Ó 243. ó 20. ▼ 52. 4 84. T 116. t 148. ω 180. ⁻¹ 212. Ô 244. ô 21. ← 53. 5 85. U 117. u 149. E 181. μ 213. Õ 245. õ 22. → 54. 6 86. V 118. v 150. e 182. ¶ 214. Ö 246. ö 23. ↑ 55. 7 87. W 119. w 151. i 183. · 215. × 247. ÷ 24. ↓ 56 8 88. X 120. x 152. r 184. + 216. Ø 248. ø 25. ◀ 57. 9 89. Y 121. y 153. T 185. ¹ 217. Ù 249. ù 26. ▶ 58. : 90. Z 122. z 154. $\bar{x}$ 186. o 218. Ú 250. ú 27. ↑ 59. ; 91. [123. { 155. $\bar{y}$ 187. » 219. Û 251. û 28. ∪ 60. < 92. \ 124. | 156. ≤ 188. d 220. Ü 252. ü 29. ∩ 61. = 93.] 125. } 157. ≠ 189. ∫ 221. Ý 253. ý 30. ⊂ 62. > 94. ^ 126. ~ 158. ≥ 190. ∞ 222. Þ 254. þ 31. ∈ 63. ? 95. _ 127. ♦ 159. ∠ 191. ¿ 223. ß 255. ÿ Page 22 68k Basic Starter Kit Welcome to the TI-Basic Developer (TI|BD) 68k Basic Starter Kit! This tutorial is designed to help new 68k programmers get their feet off the ground. The tutorial is divided into chapters that each have their own focus, and is meant to be read in sequential order. If you have questions or get stuck, leave a post on the forums and somebody will assist you. Happy coding! Page 23 Usability Imagine you are using a program for the first time. You have no prior knowledge about the program; someone just put the program on your calculator without giving you any instructions and now you are trying to figure out how to use it. After literally pressing all the buttons on the calculator and trying all sorts of key combinations, you give up and tell your friend the program was useless. You then delete the program because you figure it's worthless if you can't use it. This example isn't based off any one particular program (I don't want to name names, but more importantly talk in the general sense), but it does resonate with lots of program users who have had a similar experience. What this problem really is about is poor user-friendliness — more commonly known as usability. The definition of usability is simply how easy it is for people to use a program, and how much value it provides. While usability can take on many different forms, there are some essential things that you can do to make a program more user-friendly. In-Game Help Probably the easiest way to make a program user-friendly is by including some in-game help. While you ideally want your program to be so easy to use that a user can simply pick it up and figure out how to play it, not every game is so straightforward, and the average user probably needs some help. The best place to include help in a program is as one of the options in the program's menu. When the user comes across the menu, they will see the help option and they can select it to view the help. The help does not need to cover every minute detail about the program, but rather just explain the objective of the game and detail what keys are used for controls. :PopUp {"Option 1","Option 2","Help"},option ... :If option=3 Then : Dialog : Title "Help" : Text "The game objective is ..." : Text "Use the ENTER key to ... " : EndDlog :EndIf Because most people do not like using help unless they have to, you should try to limit your help to one or two screens at most. At the same time, if you have an extremely complex game with all sorts of features and lots of keys are needed to operate it, then it would be appropriate to include help for all of those things. The general guideline is that the amount of help needed correlates to the size of the game. Protect the User The next thing you can do to make a user-friendly program is to protect the user from themselves. Often times in a program you will want to think about what could go wrong and try to either prevent it from happening or tell the user what's wrong. Preventing it from happening involves you, the programmer, programming in safety protections for the user so that they aren't even aware that something went wrong. Say the program calls for the user to type in a number between 1-1000 and the user types in 5000. If your program just goes on with this value, it will probably crash at some point later on. Rather, it's necessary to check the value and ask for the number again if it's wrong. For example: :Input "Enter a number 1 to 1000",n :While n1000 : Disp "The number must be 1 to 1000!" : Input "Enter the number again",n :EndWhile Sometimes it might be impractical to check whether an input is valid. In that case, an alternative is to use Try..Else..EndTry blocks. If an error occurs, the program will jump to the "Else" part of the block, with the error code stored to the system variable errornum. You can either display a generic error message or try to use the error code to figure out what went wrong. Teacher Key Another part of making a user-friendly program is to include helpful features. Since the target audience is often in high school, a feature sure to be appreciated is a "teacher key." This is a special key that the user can use to quickly exit the program. When the teacher comes around, they then want to be able to get back to the home screen] so they don't get their calculator taken away. This problem is quite easy to prevent with a teacher key. In every program there is a main loop that runs throughout the life of the program. You need to add a check for whatever teacher key you want at the place in the main loop where you check for user input. Make sure, of course, that the user knows which key is the teacher key! :Loop : .... : getKey()→key : If key= : Exit : .... :EndLoop Progress Indicators In games that use maps, the program