Concepts of programming languages with advance data structures

CONCEPTS OF PROGRAMMING LANGUAGES WITH ADVANCE DATA STRUCTURES FINAL PAPER MACROS Submitted by Bustamante, Andrew Lorenzo, Erika Manarang, Paolo Marco Young, Jennifer Ms. Charibeth Cheng Faculty December 3, 2010 TABLE OF CONTENTS I. Introduction II. Scope and Limitation of the Research III. Macros a. Definition b. Usage c. Advantages and Disadvantages IV. Conclusion and Recommendation V. References I. Introduction Macros are scripts that provide a way to automate a task performed repeatedly or on a regular basis.

It involves a series of commands and actions that can be triggered by performing specific commands through the keyboard, mouse, program compilation, etc. Tasks performed by macros are commonly repetitive in nature. In effect, macros decrease the time it would take for users to do these repetitive actions in an application. These tasks can be as simple as inserting names and addresses into a word processor or as complex as launching a program, copying data from it, activating another program, pasting the data into it and repeating this several times.

II. Scope and Limitation The research will discuss the usage of macros in different programming platforms; its advantages and disadvantages. At the end of this research paper, a constructive comparison of the disadvantages and advantages will be provided. Recommendation for the use of macros derived from the given advantages and disadvantages would be part of the research scope. The programming languages that will be used for this research will include languages under the following categories: 1. Object Oriented 2. Procedural/Imperative 3.

Applications with macro programming One language will be chosen from each programming language category and 2 applications will be used for the application macros. These languages or applications were chosen because of the following reasons: 1. The researchers’ knowledge of the languages or applications 2. The current programming language trend in the society 3. The impact of the programming language to the current programming languages The following applications and languages were chosen for each category: 1. Object Oriented – C# 2. Procedural/Imperative – C . Applications with macro programming – MS Excel and Access through VB Programming III. Macros a. Definition One definition of a macro is “ a special kind of translation that may occur as the first step in the translation of a program”[1] or instructions that “ generate a word or sequence of words”[2]. However, in a more generic way, a macro can simply be defined as “ a series of commands and instructions that you group together as a single command to accomplish a task automatically”[9] These can be done by pressing a button, performing a series of keyboard commands (i. . ctrl + a + b) , etc. In programming language, a macro may be used as a way of extending the syntax of the language in a controlled way. The main purpose of macros in programming languages is to provide a way for the programmers to define a new syntax and be able to convert it to the language’s own syntax. [5] This provides flexibility in any programming language and the ease of writing programs in that language also increases. However, note that although writability in that language increases, the readability of the source code does not necessarily increase with it.

Since macros provide a way for programmers to extend the language, there is no strict rule to prevent a programmer from defining a macro which does not describe the procedure it is performing. Consider the code segment below: #define Sum (2 * 4) void main() { int i = Sum; } Reading the code segment, the expected result will be the result of adding 2 to 4 since the macro name is Sum. However, as can be seen in the macro definition, the macro Sum is not defined as adding two numbers rather it multiplies two numbers which returns a different result. This decreases the readability of the statement.

Macros are interpreted or expanded during compilation. To interpret or expand a macro during compilation means that the language translates the macro definition into the base syntax of the language and compiles the corresponding code generated from that translation. [4] Depending on the category, macros are interpreted or expanded at different parts of the compilation process. Macros can be classified into the following categories: [5] 1. Textual Macros: These macros are implemented somewhere between the lexical analyzer and the syntax analyzer. This simply replaces specific texts with values defined in another part of the source program. . Syntactic Macros: These macros are expanded somewhere between the syntax analyzer phase and the semantic analyzer or the intermediate code generator. Some examples of syntactic macros are macros which extend the syntax of a language. These can be in the form of creating a macro which will perform normal operations for that language such as loops, comparisons, etc. 3. Semantic Macros: As compared to syntactic macros, semantic macros are concerned with data types. These macros are expanded during the intermediate code generation part of the compilation.

Some examples of semantic macros are macros which converts integers to binary. 4. Computation Macros: Are expanded during the code generation phase. These macros are mostly computations which are used in the program. Computation Macros Textual Macros Syntactic Macros Semantic Macros Figure 1 shows the process that a compiler undergoes during compilation and when each categories are expanded. Figure 1. Compilation process b. Usage i. Procedural/Imperative Macros are processed during compile time. During compilation, when the code uses a macro, it performs macroexpansion.

Macroexpansion is a process that actually translates what the macro name will need to do. It expands the base syntax having the actual code of the macro call [5]. An example is shown below: 1| #include | 2| #define PI 3. 141617| 3| | 4| void main()| 5| {| 6| printf("%. 4f', PI);| 7| }| Figure 1 Base Syntax 1| #include | 2| #define PI 3. 141617| 3| | 4| void main()| 5| {| 6| printf("%. 4f', 3. 141617);| 7| }| Figure 2 Expanded Code after Compilation The C pre-processor, often known as cpp, is a macro processor that transforms the code before compilation.

This is automatically called by the C compiler. C pre-processor is called a macro processor because it allows definition of macros. [6] The pre-processor performs a series of textual transformations one of which is expanding the macros. Once these transformations are completed, the code is passed to the compiler. [6] Constants in C can either be declared via macros or by variable declaration and value assignment. As for macro definition of constants, sample syntax is the one seen in line 2 of figure 2. When a constant is declared via macro definition, all names rooting to that macro is textually substituted before compilation.

On the other hand, constants declared via variable declaration are included in compilation. The advantage of macro constants is that it can be used to define sizes of arrays and can be used for switching statements as compared to macro constants. However, despite the advantage variable-based constants are still identified as variables during debugging activities as compared to macros that cannot be identified anymore. [2] ii. Object-Oriented Unlike imperative languages such as C, macros are less used in object oriented languages.

This may be because object oriented programming provides a way to simplify coding without using macros. An example of this is the declaration of enumeration or constant variables in object oriented. In C#, it is possible to have an enumeration class which works exactly like textual macros. Consider the code segment below: Enum class: Usage: Result: Notice how the enum value was used to represent the integer 1. Comparing it to textual macros of C, they are both performing the same usage. Except that in C we define the variable zero as a macro in the same class instead of creating a new file which will contain only enum values.

Alternatively, to be able to simulate macro-like process in C# there are developers who have developed a new process or compiler to be able to do this. One such process is the CS Script process developed by Oleg Shilo [7].

His idea is to extend the MSBuild process to recognize specific syntax which will be run during compile time. An example of this process is shown below. In the given example, notice that in the Test class there is what seems like a commented out code which calls for the generator class.

This part of the code is the syntax of the CS Script to run and instantiate a class during compile time. Note that all calls are done at compile time. [6] This is a somewhat basic example but since this is already in C# there are a lot more possibilities that may be considered. Generator class Test Class Result MSBuildCommunityTask, an extended version of Visual Studio’s MSBuild, also provides a somewhat similar functionality but through tokens replacement. Since C# does not allow macro processing, the MSBuild process was extended to simulate the process of textual replacement.

This however poses a problem of knowledge of the MSBuild process since to be able to fully utilize the capabilities of MSBuild, new code segments should be created which extends to the MSBuild project. Chiba[9] also introduced a new process to implement actual object oriented macros in C++. His idea was that regular macros lacked the capability to handle the complexity of object oriented languages therefore do not properly support the needs of object oriented languages in terms of macro processing. He developed an extended version of OpenC++ which makes use of metaobjects to create macros. 10] However, OpenC++ can only be run and compiled using Linux, Solaris or Unix systems. iii. Macro Programming in Excel Macros are powerful tools in all MS/Office applications (including Word, Excel, etc.) that let users record keystrokes for later playback. These are collections of visual basic for application (VBA, a simplified version of Visual Basic) code that allows the application (in this case Inventor) to perform complex actions automatically with limited user input. Tasks that would normally require many keystrokes and/or mouse clicks can be recorded for later playback with a single click.

Once the keystrokes (and/or mouse clicks) are recorded, the recording (called a macro) can be edited and fine-tuned until it implements the exact procedure that the user would want to be implemented. After the macro is complete, it can be assigned to a new button on the toolbar, or to a key on the keyboard, thus allowing it to be played back with a single click. An example of a macro might be a program that inserts multiple parts into an assembly document or a simple toolbar button that accesses menu commands that does not have a default toolbar button.

In Excel, macros facilitate demonstrating the need for repetitive analysis in numerical techniques. A macro can also do conditional activities. For example, if a selected text is blue then color the text red, otherwise/else color it blue. The undo command (control + z), however, does not always work with macros, therefore, during creation of macros and before implementing the macros, the user has to make sure that the macro will perform the task that needs to be performed. By default macros are disabled in any MS Office applications, therefore, to use a macro, this feature needs to be enabled first. . Launch Excel, Word, or any other Microsoft Office application. 2. Click – – – Tools | Macro | Security. 3. Click on the security tab and set your security level to medium. New versions of MS Office allow users to “ record” macros by pressing the record button provided in the application. This provides a way for users to use macros without coding the exact steps that the macros has to take. The steps to create this macro is specified below. 1. Launch Excel, Word, or any other MS/Office application. 2. Click – – – Tools | Macro | Record New Macro. 3.

Select a place to store it via the “ Store Macro In – – – “ selection box: a. Global: Available to all documents b. Local: Restricted to the current document and copies made from it. 4. Option: You may assign a shortcut key (control + __). 5. Click OK to begin recording keystrokes. 6. IMPORTANT: When done remember to click ” STOP RECORDING”. To run saved macros, the user can choose to run it by performing the specified key stroke, pressing the macro button or going to the tools menu to run the macro. These steps are enumerated below. 1. Select – – – Tools | Macro | Macros … 2.

Use its shortcut key(s) – – (Control + __) (if you have assigned it to a shortcut) 3. Create a button (or other control) for it on your document 4. Program it to be auto-activated by an event or timer 5. Create a new button for it in one of your toolbars 6. Create a menu item for it on an existing or new drop-down menu 7. Add it to a right-click menu (a context-sensitive menu) triggered by a right-click event 8. Tie it to another event such as a. Moving the cursor b. Selecting a different cell (in Excel) c. Moving to a new line (in Word) d. Opening or closing a file e.

Having no activity for a specified time period f. Hovering over (moving the mouse over) a particular area of a document g. There are many other events that can be used as triggers for macros. A recorded macro can also be edited by going to the tools menu and choosing to edit a specific macro. Editing a macro will be done through VBA coding. iv. Macro Programming in Access

Macros in Access can be considered a simplified programming language that can be used to increase the functionality of the database. Commands are pre-written in a language called Microsoft Visual Basic for Applications, referred to as VBA.

The skills needed to record and write macros are surprisingly easy to learn because of what the user already knows just by using the program. Example, it can enclose a macro to a command button in a form, so the macro is executed when clicking the button. The macros contain actions that make tasks, like opening a report, executing a query or to close the database. Almost all the operations of databases that normally are made manually can be automated through macros. In MS Access 2003 the only way to create macros is to program it by using the VBA language incorporated in all MS Office applications.

It is one of the more easy-to-learn programming languages and it does not require the complex programming techniques that are necessary to program C++ or other high level languages. VBA also provides a user-friendly, form-based interface to assign variables and simplify code development. VBA is used to do any of the [8]: 1. Create a new function. 2. Create a new subroutine. 3. Define a global variable. 4. Place code behind an event procedure such as the “ On Click” event of a command button. 5. Execute the Run Code action in a macro. The user interface of the macros interface in MS Access is shown in the image below:

Below is a sample code of coding in VBA in Access. The below code will create an MDB file and imports 2 tables from another MDB file to the newly created database. Sub CreateNewMDBFile() Dim ws As Workspace Dim db As Database Dim LFilename As String ‘ Get default Workspace Set ws = DBEngine. Workspaces(0) ‘ Path and file name for new mdb file LFilename = “ c: NewDB. mdb” ‘ Make sure there isn’t already a file with the name of the new database If Dir(LFilename) <> “” Then Kill LFilename ‘ Create a new mdb file Set db = ws. CreateDatabase(LFilename, dbLangGeneral) For lookup tables, export both table definition and data to new mdb file DoCmd. TransferDatabase acExport, “ Microsoft Access”, LFilename, acTable, “ Lookup Table1”, “ Lookup Table1”, False ‘ For data entry tables, export only table definition to new mdb file DoCmd. TransferDatabase acExport, “ Microsoft Access”, LFilename, acTable, “ DataEntry Table1”, “ DataEntry Table1”, True db. Close Set db = Nothing End Sub In MS Access 2007, Microsoft has simplified the creation of macros by providing an interface for creating macros instead of programming it in VBA. A sample of this is shown below.

The below sample creates a new form in MS Access. 1. In the Create tab, in the group Other, click in Macro command, then Macro menu item. 2. The Macro definition panel will appear. 3. Select the OpenForm action. 4. Configure the OpenForm arguments. 5. A condition specifies certain criteria that must be met before an action will be performed. You can use any expression to evaluate to true or false or Yes or No the expression and execute or not some commands based in these results. 6. Once the action and arguments are configured, save the macro. 7. Open a Blank Form in Design View. 8. Expand the Form Footer area. 9.

Select the Command Button control from the Controls Tab. 10. Draw the button at the Form Footer area. 11. Select the Miscellaneous and Run macro Options from the emerging menu. Press the NEXT button. 12. Select the previously saved macro. Press the NEXT button. 13. Assign a text or an image to the button. Press the NEXT button. 14. Assign a name to the button control. Press the FINISH button. 15. Save the form and test it. c. Advantage and Disadvantage Macros have a lot of advantages. To summarize Kiselov's statement[11], the main advantage of macros is to provide a way for programmers to have easier access to certain procedures.

These procedures may be conditional statements, loops, or simple textual representation. This advantage is the same for all programming languages or applications. As discussed in the previous sections of this paper, macros help in automating repetitive procedures. However, with these advantages, there are also disadvantages in using macros. Kiselov[11] mentions some of the disadvantages of using macros in programming language in his paper. These disadvantages are enumerated below: 1. Rules governing macro expansion: A macro might be using a variable name which is also used locally by a specific method. . Complexity of Macros: Since macros are complex by nature, they are difficult to debug and handle 3. Subtle semantic error: Some very common semantic errors that may be caused by macros, as mentioned by Stallman and Weinberg[3] are swallowing of semicolon, misnesting, operator precedence, duplication of side effects, and self-referential macros. Macros created to automate tasks for applications such as MS Excel and MS Access are very useful to ensure that the process will always output the same result regardless of the number of times it is run.

This helps lessen errors caused by performing a repetitive task manually. However, these macros, although useful, also has some disadvantages. These disadvantages are enumerated below. [3] 1. Knowledge in programming is needed: To use a macro for MS applications, even a little knowledge of programming is needed since MS macros use VBA language. 2. Limited to the application where the macro was developed: The VBA codes cannot be taken out of the application that is using them. This means that if the same code is needed but in a different application or machine, the macro must be recoded. . Troubleshooting: If there is a problem in the macro, the VBA script must be debugged to fix the issue. Bottom of Form IV. Conclusion and Recommendation Macros are very powerful tools if used properly. They can simplify the manual task that users are normally subjected to, whether when programming or using MS Office applications. For programming languages, the use of macros is useful since programmers tend to use a lot of the same values in different files or different functions. Consider having a variable which identifies where a specific file should be retrieved.

If this variable is declared locally multiple times in different files or different functions or methods there is a high probability that the programmer might neglect to change one of them which would render the system incorrect. For application macros on the other hand, macros simply make the user’s life easier. Especially when using excel where users tend to use multiple sheets in a single file where all sheets are related with each other. Macros can help automate the task of updating each sheet if a value is changed in one sheet with just a press of a button.

However, despite all these advantages, there are also a lot of disadvantages in using macros as mentioned in the previous section. These disadvantages can be avoided by simply using the macros in the proper way. For programming language some recommended usage of macros are enumerated below: 1. Use parenthesis for arithmetic operations: The use of parenthesis in macro definition groups the result of compound arguments.[11] 2. Completion of expressions' syntax in macro definition: Since macro is textually substituted to the base syntax, it is considered a best-practice that the whole body of the macro definition be complete[10] 3.

Exclusion of semicolon for function-like macro calls[11] 4. Pre-computation of macro arguments: For macro implementation that invokes function calls. The functions to use in the macro definition should be computed first. Repeating computation within macros should be eliminated to minimize the risk of multiplying the error that might be brought by the function calls[11] 5. Make macros as simple as possible: Macros are already hard to debug and troubleshoot as is, having a very complex macro makes this process harder and more error prone than usual. 6.

Analyze processes that should be made into a macro: Especially for object oriented programming where there is a do a code where a method may act the same way as having a macro, if a macro is not needed then don't use it. As for MS application macros, the best recommendation that the proponents are able to give is to create macros for specific constant tasks. If a process tends to change very often, then this process should not be made into a macro. This recommendation prevents the constant modification of the macros defined in the applications and lessens the probability of having errors due to the modification.

Despite having the proposed correct implementation, the enumerated actions might not cover all scenarios on implementing algorithms. The enumerated correct implementation was based on numerical computation and might need to be tested further on highly sophisticated macro definitions. V. References Books 1. Jazayeri, M. and Ghezzi, C. Programming Language Concepts. [1998]. John Wiley & Sons, Inc. 2. Johnsonbaugh, R. and Martin K. [1997] C for Scientists and Engineers. NJ: Prentice Hall. 3. Walkenbach, J. [2004]. Excel VBA Programming for Dummies. NJ: Wiley Publishing Online 4. Microsoft Corporation. 2010). Visual Studio Macros. [online]. Available: http://msdn. microsoft. com/en-us/library/b4c73967%28VS. 80%29. aspx 5. Gabriels, R. and Gerrits, D. [2005]. A Comparison of macro systems for extending programming languages. [online]. Available: http://dirkgerrits. com/wp-content/uploads/avp-essay. pdf 6. Stallman, R. M. , and Weinberg, Z. [2010] The C Preprocessror. [online]. Available: http://gcc. gnu. org/onlinedocs/gcc-4. 2. 4/cpp. pdf. 7. Shilo, O. [2010]. C# Script Execution Engine. [online]. Available: http://www. csscript. net/CurrentRelease. html 8. Tech on the Net (2010).

MS Access Tutorial : VBA Basics in Access. [online]. Available: http://www. techonthenet. com/access/tutorials/vbabasics/basics01. php Journals 9. Greenwald, I. D, Kane, M. [1959] The Share 709 System: Programming and Modifications. Journal of the ACM. Volume 6. Issue 2. p123 – 127. 10. Chiba, S. [1998]. Macro Processing in Object-Oriented Languages. Technology of

Object Oriented Languages. Tools 28. P133-126. 11. Kiselyov, O. [2002]. Macros that Compose: Systematic Macro Programming. Proceeding GPCE ’02 Proceedings of the 1st ACM SIGPLAN/SIGSOFT conference on Generative Programming and Component Engineering. P202-217