European Journal of Business and Social Sciences, Vol. 3, No.3 , pp 178-188, **June 2014.** **P.P. 178 - 188**
URL: *http://www.ejbss.com/recent.aspx*
ISSN: 2235 -767X

A STRING SEARCH MARKETING APPLICATION USING MIT APP INVENTOR VISUAL PROGRAMMING

Jerry Chin
Professor and Department Head
CIS Department
Missouri State University
901 S National
Springfield, MO 65897
417-836-4131
Jerrychin@missouristate.edu

Mary Chin
Senior Instructor
Marketing Department
Missouri State University
901 S National
Springfield, MO 65897
417-836-4873
Marychin@missouristate.edu

ABSTRACT

Using the App Inventor software, this paper extends a previous paper that demonstrated the use of programing software that provides the student programmer visual cues to construct the code for a student programming assignment. This method does not disregard or minimize the syntax or required logical constructs. The student can concentrate more on the logic and less on the language itself.

Key Words: Graphical Programming, Introductory Programming, String Search Application

The MIT App Inventor [1] development system is Java based and incorporates a web browser and for this paper, an Android phone emulator. The student stores his/her work on the App Inventor servers. Programming occurs in two phases. In the Designer screen, the student selects and inserts components such as buttons, textboxes, and labels. The Blocks Editor provides lists of colored blocks that interact with each other depending upon the function. Blocks for logic,control, and variables are assembled to provide the program code.

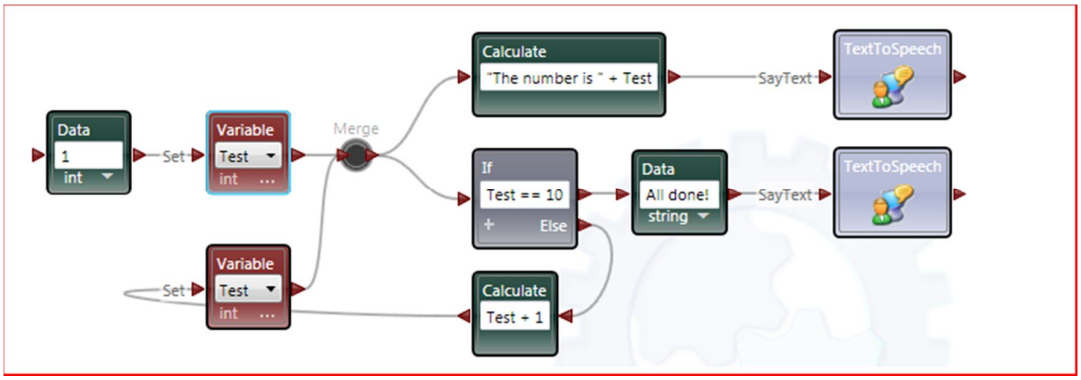
From a teaching and learning standpoint, the syntax issue is essentially taken out of the development. When incompatible blocks are joined to together, the action is blocked with an error pop-up. The logic, analysis and arrangement of procedures still remain the student's perogative.

Visual Programming

In addition to the App Inventor software, there are a number of software packages that can be classified as visual programming languages. We list three other examples which are free downloads.

Programming Without Coding Technology [2] is a software package that provides a GUI, which allows the student to construct a program by using an interface with pop-up boxes and templates to reduce the need to know the exact syntax. PWCT is a free download. The student is still required to have a logical overview of the code. For the problem presented in this paper, the string functions provided were sufficient. The GUI provides a structured way to control the levels of code such as code inserted within a While-Loop or an If-Then statement.

Microsoft VPL [4] is a graphical programming package that is part of the Microsoft Robotics Developer Studio. This application is used to guide and operate robots. The approach of software uses a graphical dataflow-based programming model, where the components initiate action once all the signals or messages arrive rather than a sequence of commands executed in sequence. The language purportedly targets the beginning programmer. The following picture shows the code to increment a variable by one until it reaches a maximum of ten.

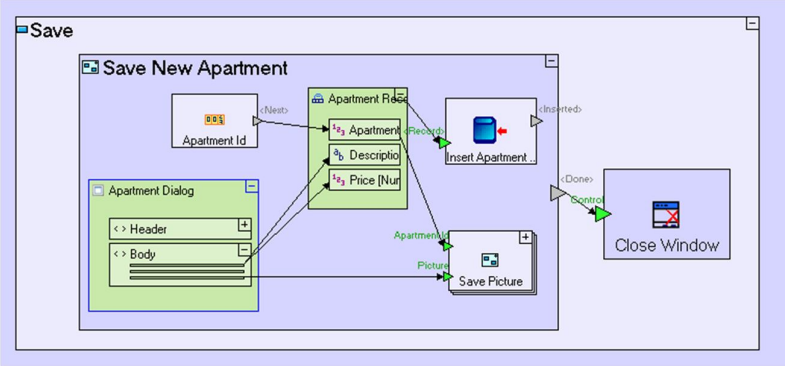


Microsoft VPL Code

Increase Counter to 10

The programmer chooses from a list of services or functions, which are dragged and dropped on a diagram window. By opening the graphical representations, the programmer is prompted to supply the name of values or variables when needed. Connections are easily made by linking output nodes to input nodes. As can be seen in the example, inputs can be merged when appropriate.

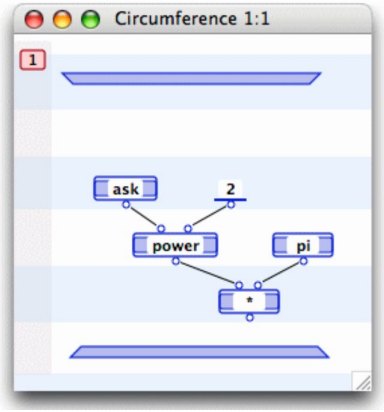
Tersus [3] is a language based on drawing graphical diagrams rather than writing statements. The following picture displays an example.



Tersus Code Example

In the diagram, clicking on a component enlarges the graphic so that other components can be inserted. In this way, a simple flow into a component can be changed as future components are layered within the component.

Marten [7] is based on Prograph [6] and is currently available at the Web site, http://andescotia.com/support/. Marten's approach is the insertion of graphics which are connected via data links. Entry terminals set values and output roots to pass on results. A sample program for the area of a circle is depicted in the following figure. The "power" block has two entry roots and one output root.



Marten Code Example

The Student Problem

This student problem is based upon the truth-functional form algorithm found in the study of first order logic. FOL is a study of sentences and the logical consequences when these sentences are considered in a specific framework. In particular, the algorithm boils down to a string search and replace problem [5].
Eastern Imports has a legacy distribution system where internal records show the manufacturing plant, item and destination. Because of the competitive marketing conditions, customers have asked for more confidentiality of their business purchases. Eastern has decided to code their invoices when external intermediary shippers are involved. Plants and destination address are all that are revealed in Table 1. The system should also provide the capability of decoding to reproduce the original internal record.

Table 1

Plant	Distribution Warehouse
D = Dallas	L = Los Angeles
A = Atlanta	N = New York City
DxLx AxNx	Ship x from Dallas to Los Angeles Ship x from Atlanta to NYC
Ny	Pick up y from New York City
Lw	Pick up w from Los Angeles
Dx(note) Ay(note)	Dallas special order for x Atlanta special order for y

Logic of the Search

Upon inspection, there are three possible forms of substrings given the table above. We list them in Table 2 as the following:

Table 2

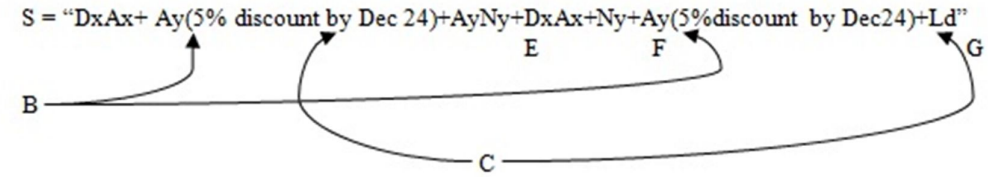
F1: e.g. Lx, Ny
F2: e.g. DxLx, AyNy
F3: e.g. Ay(5% discount by Dec 24)

Strings are to be transformed by a series of string operations. The search starts from the left, where the index is the position of the string character at 1, namely S_1. We will also check the character at S_3. Depending upon the value of S_i, we know the substring under investigation is of the three distinct forms, as indicated in Table 3.

Table 3

S_i	S_{i+1}	S_{i+2}	S_{i+3}	Form	Length
A	x	N	x	F2	4
D	X	(		F3, where S_j = ")"	j-i+1
L	y			F1	2

A message in the system might look like:



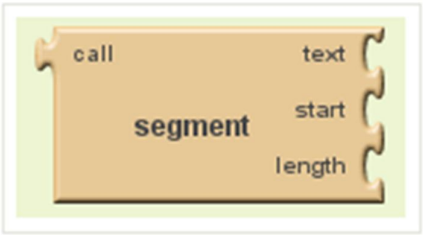
In general, the algorithm searches the string beginning on the left and replaces matching substrings by a unique single character. For example, suppose that the set of substitution characters are the letters {B,C,E,F,G,H}. Then the transformed string S becomes S = "B+C+E+B+F+C+G".

Note that B and C are repeated twice, indicating a multiple occurrences of two substrings, "DxAx" and "Ay(5% discount by Dec 24)"which were replaced by "B" and "C", respectively. The string S has been transformed into a simpler string while preserving the general structure of S.

Preview of App Inventor

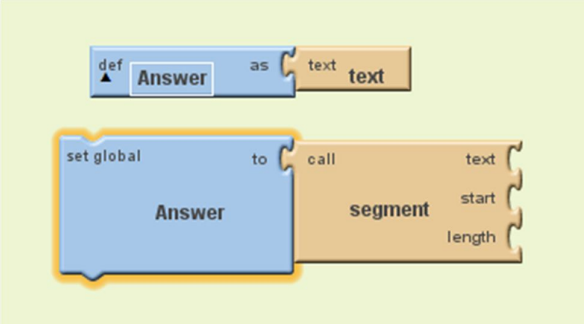
Since the string is a main structure of the problem, a quick view using the App Inventor approach follows. Suppose a string with n characters looks like $S = s_1 s_2 s_3 \ldots s_i s_{i+1}, \ldots s_m \ldots s_n$. A substring beginning at starting position index i and ending at m has length m-i+1.

The segment code block requires string S, index i, and length = m-i+1. When length is equal to one, then a single character can be extracted from a string and examined. By altering the start value and keeping the length equal to one, then a character at any position or index can be selected. When the length is greater than one, then a substring is being studied..



Segment Block

The programmer is prompted by the block. If the programmer attempts to connect a numerical value to the text slot, the system error message message will pop up. The result is assigned by connecting the segment block.



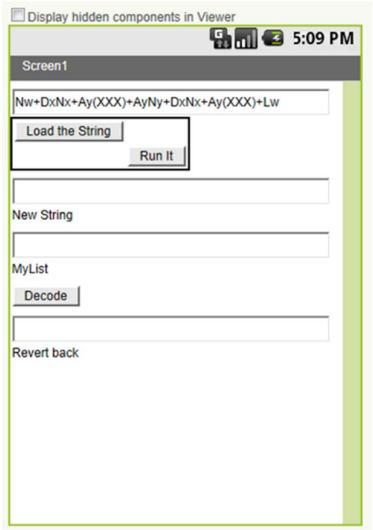
Assignment Block

Answer is first defined as a text variable and then assigned the result of the segment block. One can now see how the programming task is more logic and analysis. The segment code block is used throughout the App Inventor code examples below.

The Algorithm

The rationale for Table 3 follows. Let S_i be the character string at the i^{th} position in string S. If S_i is a substring of "DALN", then we can search the string to determine if the form is F1, F2, or F3. Otherwise, bump in the index of S to i+1 and look at S_{i+1} . If S_i = "A" or "D", then we have F2 or F3. Check S_{i+2}. If S_{i+2} = "L" or "N", the form is F2 and its length is 4. If S_{i+2} = "(" then the form is F3. At this point, we must determine j such that S_j = ")". The length of the F3 substring is j-1+1. If S_{i+2} is not a substring of "LN(" then by default the form is F1 and its length is 2.

To create an application, the student is presented with the Designer which provides a blank viewer. The student drags components from a palette of components to the viewer. In our example, we used four textboxs, two buttons and indentifying labels. The string is pre-loaded as the default value of the textbox. When the button labeled, "Load the String " is depressed, the string shows in the textbox. Then The "Run It" button executes the app and the answer appears in the second texbox, labeled as New String. To get back the original string S, the button "Decode" is depressed and S should appear in the last and fourth textbox. The significance of the third textbox MyList is discussed below.

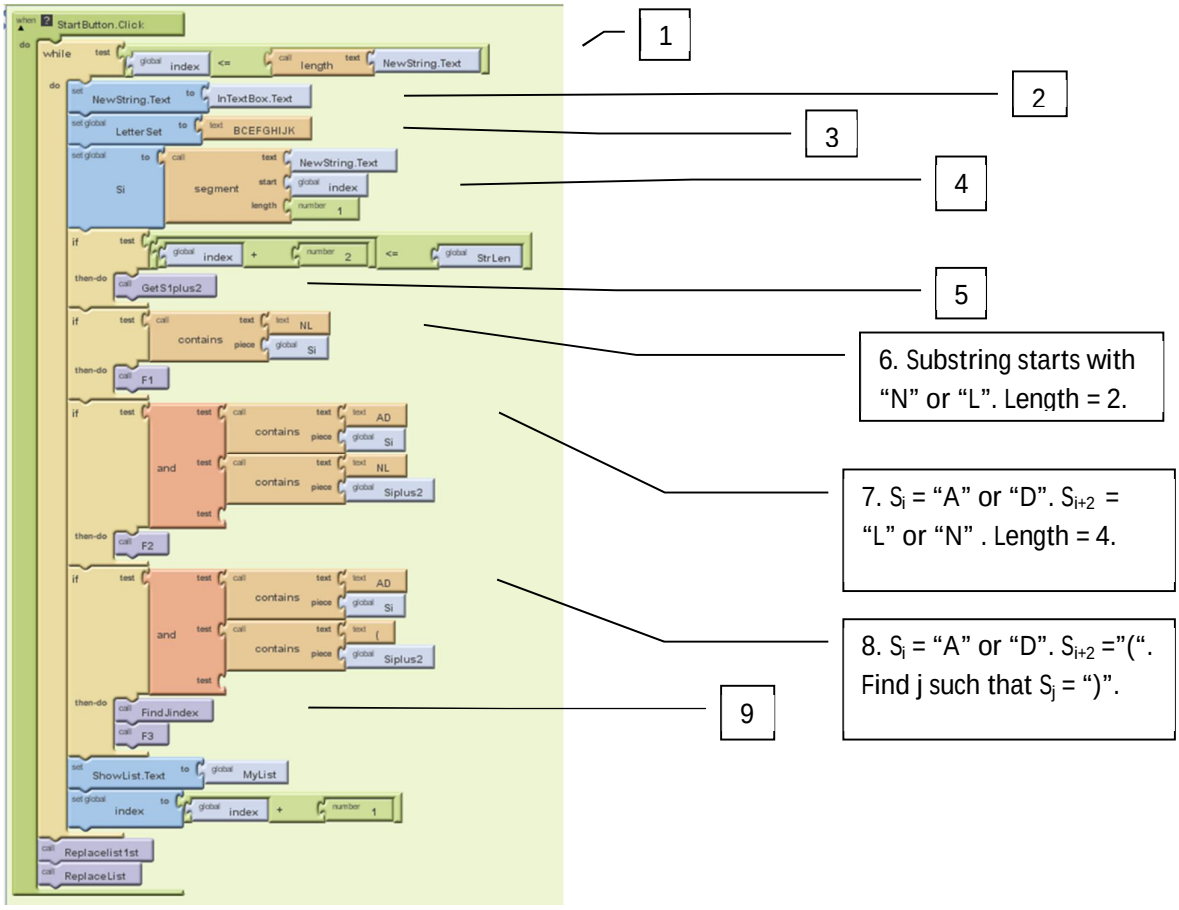


TheViewer

For this discussion, the code blocks are numbered in figure below called Start Block. A particular block's number will be displayed like "(5)".

The first task is to begin the application. The StartButton is depressed and the While control block is in control. The counter for the string is a variable called index. The While loop (1) will be in control as long as index is less than or equal to the length of the string in question. The pre-loaded string S will be read and inserted into the first textbox at (2). The Letter set is located in (3). Letter set holds the assigned letters that will be used to replace substrings within S.

If the index = i, each character in S, a substring of length 1, is assigned to S_i in the code blocks (4). If the string is long enough, call a procedure called GetS1Plus2, resulting in the character at position i+2 . Call this character S_{i+2} . (5)



Start Block

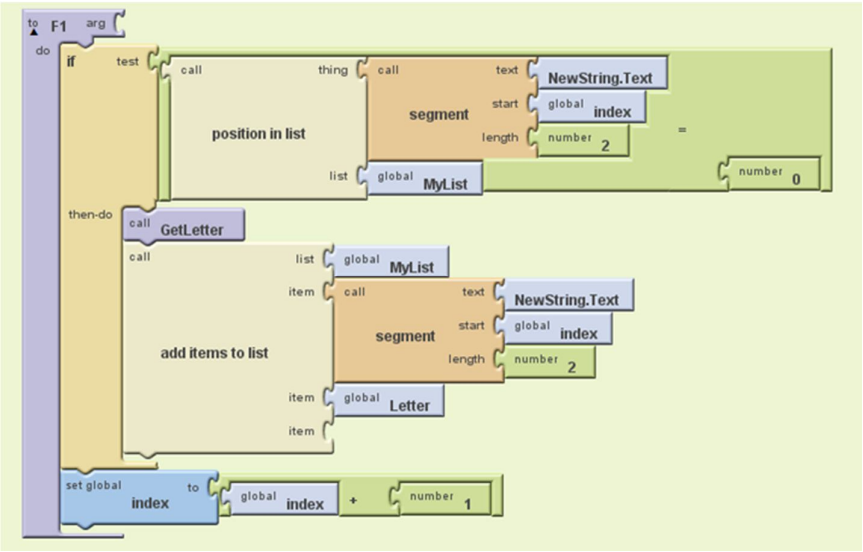
Code Blocks for Form 1, Form 2, and Form 3

The first procedure (6) we cover is for the F1 form. MyList is a list composed of a substring and its unique letter assigned from {B,C,E,F,G,H}. If a substring appears more than once in string S, then its corresponding letter substitute is always the same. In our example above, the letter for DxAx is the letter B. The substring DxAx appears twice in S and the one letter B is used twice as a substitute.

F1 Procedure

F1 forms are composed of two characters and begin with the letter L or N. The procedure first checks to see if a substring, F-string, already exists on MyList. Segment is a string operation that required a text string, a position to start, and a length. In our particular example, F-string of NewString.Text beginning at index for a length of 2 is searched in MyList. If the answer is no, then the "position in list" is zero. If the F-string does not exist on MyList, then GetLetter is called to get a new letter. The next groups of blocks add F-string along with its corresponding Letter to MyList. If the position-in-list value is not zero, then F-string has been processed before. In either case, bump the index of S and process the next character in NewString.Text

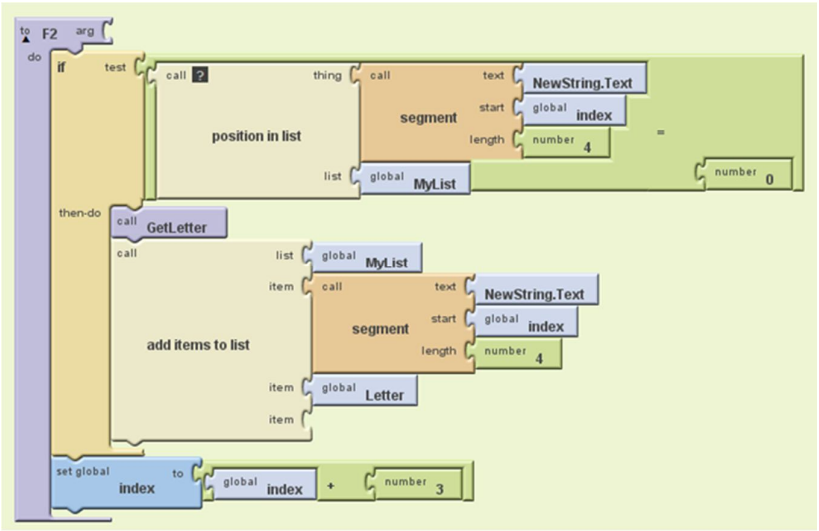
Note that the code is composed of interlocking colored blocks to indicate the purpose of the block. The shapes and colors of the code blocks aid the programmer in the construction of the code. Errors and syntax cues problems are minimized by this scheme.



F1 Code Block

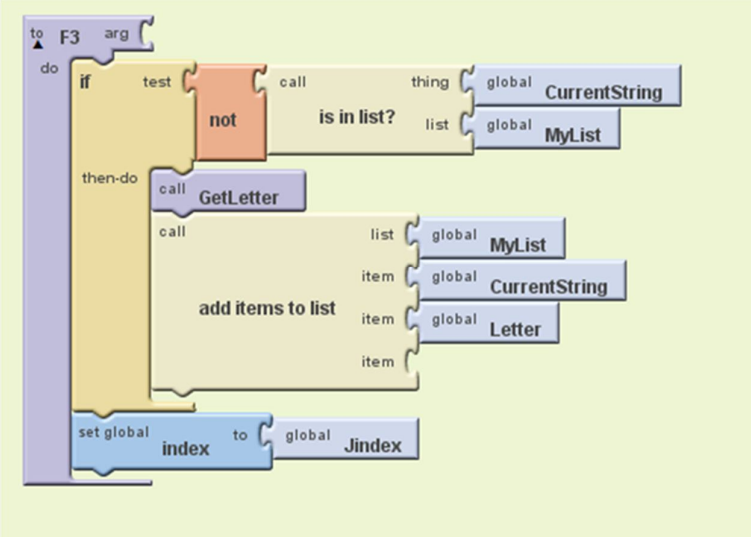
F2 Procedure

In the F2 procedure, the F-string has length 4 and begins with a first character of either A or D (7). Like before, if F-string is not on MyList, then a new letter is selected and added to the list with F-string. Inspection of the code blocks shows a length of 4.



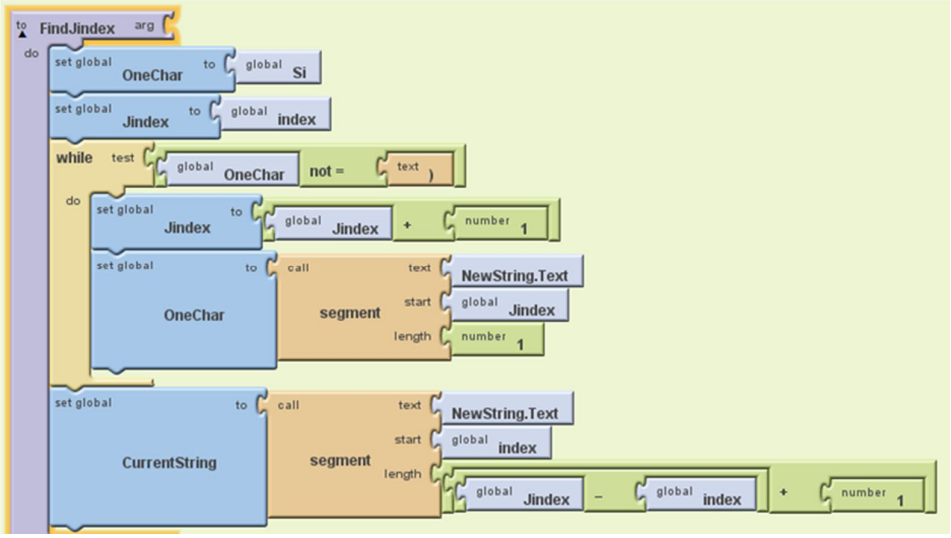
F3 Procedure

The F3 code for a particular F-string does not have a hard coded length as in the previous two procedures (8). A F3 string begins as either an "A" or a "D" at character Si and two positions down the string has a left parentheses, "(" in the character, S_{i+2}. The code before the F3 call finds the position of the closing right parentheses, S_j = ")".



F3 Code Block

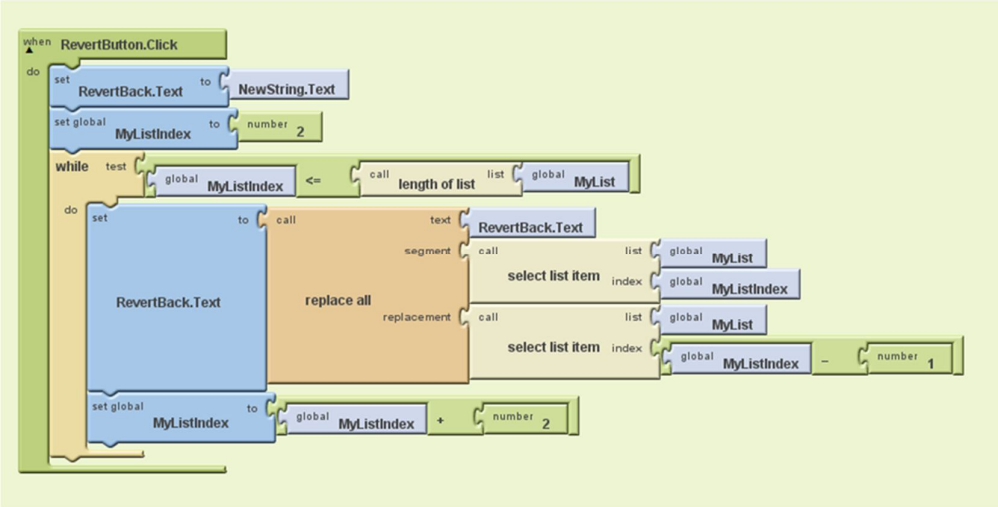
This is accomplished in a procedure, called FindJindex (9). The code block is shown below. Once that determining character has been found the F-string can be determined and is called CurrentString.



FindJindex Code Block

Decode Procedure

The last procedure is to decode the String of letters back to the original string S. The procedure is called by depressing the Revert Button which calls the RevertButton.Click function. The structure of MyList is {S_1,B,S_2,C,S_3,E,….}. So, we load the search index with 2, pointing to "B". Use that "B" and replace all occurrences of "B" with S1. Similarly, replace all occurrences of "C" with S_2 . By bumping the MyListIndex in the While loop, MyList is processed. The final string result is assigned to the last textbox.



Decode Procedure

Conclusion

The PWCT software still requires that the user have a familiarity of computer structures such as the While-Loop and the If-Then. This software shows the user to some extent how to formulate the code. Moreover, there is the expectation of procedures being defined and called along with accompanying parameter lists and returned values. Lists were used to store and provide values. Syntax errors are decreased since the software prompts the user. The user provided the variables and the logic to solve the problem. For the beginner, this system lets the user concentrate on the problem and the programming logic.

Bibliography

1. MIT App Inventor. [Computer Software] Available at http://appinventor.mit.edu/explore/
2. Programming Without Code Technology. PWCT. Version 1.7 (Sharp) 2010.09.15. Available at http://doublesvsoop.sourceforge.net/
3. Tersus. [Software] Available at http://www.tersus.com/
4. Microsoft VPL [Software] Available at http://msdn.microsoft.com/en-us/library/bb905471.aspx
5. Barwise, J. & Etchemendy, J. Language Proof and Logic. Stanford, CA : CSLI Publications, 2003.
6. Steinman S. & Carver, K. Visual programming with Prograph CPX. Greenwich, CT: Manning Publications, 1995.
7. Marten. [Software] Available at http://andescotia.com/support/