University of Helsinki
Department of Computer Science
Series of Publication C, No. C-2000-52

Flowchart programming with Excel

Jukka Eskola and Jorma Tarhio

Helsinki, October 2000

Report C-2000-52

Department of Computer Science
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland

ISBN 951-45-9535-1
ISSN 1457-7070

Flowchart programming with Excel

Jukka Eskola*
Department of Computer Science
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki
Finland
jueskola@cs.helsinki.fi

Jorma Tarhio†
Department of Computer Science
University of Joensuu
P.O. Box 111, FIN-80101, Joensuu
Finland
tarhio@cs.joensuu.fi

Abstract

We show how to implement a flowchart programming environment with Microsoft Excel using visualization and macro programming features of Excel. Our system, called Flopex, visualizes the execution of a program, and it can be utilized in teaching programming. Advanced features of Excel can directly be accessed from the flowchart program, and so even the animation of user data structures is possible. Flowchart programming is not our aim in itself, but we demonstrate the capabilities of Excel as a platform for implementation of visual languages and making experiments with them.

Computing Reviews (1998) Categories and Subject Descriptors:

D.1.7 Visual Programming

D.2.2 Design Tools and Techniques: Flowcharts

General terms: program visualization

Key Words and Phrases: flowchart programming, animation of flowcharts

*Corresponding author

†Current address: Department of Computer Science, Helsinki University of Technology, P.O. Box 5400, FIN-02015 HUT, Finland.

1 Introduction

Traditionally, Microsoft Excel is only seen as a spreadsheet program, but its macro language *Visual Basic for Applications* (VBA) opens new dimensions and uses, because VBA provides an unusual and fascinating programming environment which integrates code, graphics, and spreadsheet functions. To name an example, Excel can be used to visualize and illustrate various phenomena [4], and it is even suitable for animating algorithms [6].

There are several ways and techniques for presenting algorithms — especially flowcharts have been popular since the early days of computing. They have been found to promote the understanding of the structure and functionality of algorithms [7, 1], and therefore flowcharts are well suited to teaching traditional programming. Flowcharts can also be used to examine how a program should work. On an abstract level, a flowchart representation offers the same mechanisms available in traditional programming, such as sequentiality, selection and repetition. Thus, a flowchart can be interpreted as a program or as a static visualization of a program.

In this paper, we demonstrate the capabilities of Excel as a platform for versatile visual programming environments. We introduce a flowchart programming system called Flopex (FLOwchart Programming with EXcel). Flopex visualizes the execution of a flowchart, which is constructed with ready-made flowchart icons. The interpreter written in VBA executes the flowchart program interactively. Flopex is currently at the level of experimental prototype, and our plan is to make it publicly available later on.

Flopex is not just another visual system for traditional flowchart programming with icons, but it is a tool which integrates Excel with a visual program, because virtually all the functions and features of Excel may be accessed from a flowchart program. This will bring additional value and uses for flowchart programming. To name an example, one can utilize the searching functions of Excel in a flowchart program to find a certain record in a database. As a byproduct, Flopex is even able to show simple animations on the data structures used in a flowchart program.

Flowchart programming is not the aim of this paper in itself, but we demonstrate the capabilities of Excel as a platform for implementation of visual languages and making experiments with them.

The rest of the paper is organized as follows. Section 2 contains background information about visualization features of Excel and flowchart programming. Section 2 also briefly introduces some iconic programming environments. Section 3 presents how to use Flopex. Section 4 describes some implementation details of the system. Before the concluding remarks (Section 6), some characteristics of Flopex and their impacts are briefly presented and discussed in Section 5.

2 Background

2.1 Flowchart Programming

An algorithm is a sequence of instructions for solving a specific task. Algorithms are usually written in a natural language, in a programming language, or in an intermediate form called pseudocode. Besides textual representations, algorithms can also be presented as flowcharts. Flowcharts have been virtually the only widely-used visual tool programmers have had for designing programs. The flowchart representation is good for beginners, because it is illustrative and intuitive, and it releases the user from the tricky details of the syntax rules.

Flowcharts are commonly used to depict how a program will work to support comprehending its operations. Flowchart symbols represent these operations. Basic symbols are an oval (start and end), a rectangle (action), and a diamond (conditional branch). Symbols are connected by directed lines indicating the control flow through the diagram.

Often, when people speak of flowcharting, they think of a programmer tracing the logic path of an algorithm. But a flowchart can represent more functions than the plain logic path of a program. It is a clear graphic representation of a process, regardless of whether that process is an algorithm or a manufacturing procedure. Creating a flowchart is the first step towards managing the process. Originally, flowcharts were hand-drawn, which makes changes difficult. As computers became capable of graphic representation, flowcharting software was at first developed merely for programmers.

There are several iconic programming environments for teaching programming with flowcharting. Calloni and Bagert [1] have developed an environment called BACCII++ for teaching procedural and object-oriented programming. In BACCII++ all program constructs including variables are represented as icons. The user selects an icon (a statement) from the tool palette and then places it on the coding screen. BACCII++ is able to generate source programs in several programming languages from the flowchart, but the flowchart itself is not executed.

Ziegler and Crews [8] present an integrated environment called FLINT (FLowchart INTerpreter) for teaching top-down design, algorithm development, testing, and debugging. FLINT supports building algorithms as flowcharts and executing them interactively with a flowchart interpreter.

Scanlan [7] showed that approximately 75% of his students comprehended algorithms better when using flowcharts than textual representation. Therefore it is natural to apply flowchart programming in teaching.

2.2 Excel and Visualization

Spreadsheet programs like Microsoft Excel are usually regarded as tools for financial calculations [5]. This limited viewpoint restricts their use, although they offer efficient methods to visualize, illustrate and describe many kinds of phenomena and make experiments with these. According to Dybdal *et al.* [2, 3], even the use of basic spreadsheet operations supports problem-oriented, discussing, and experimenting learning. In addition to this, Excel also offers a large collection of visualization features, even for implementing animations. Microsoft has gradually added new features to Excel, and therefore the potential of the current Excel for uncommon uses is not widely known.

Macro programming is a feature that enhances the expressive power of Excel, and makes Excel a flexible working environment for implementing various visualizations. The Excel macro language, *Visual Basic for Applications* (VBA), is a programming language similar to Visual Basic, but its programming environment is slightly different. VBA, which has been the macro language of Excel since version 5.0, contains the most common features of programming languages, including standard control structures, recursion, and a wide range of library functions. Each activity or function of Excel has a command or expression in VBA, which multiplies the possibilities of using Excel in visualization because the activities can be called and controlled by a program. A fascinating feature is that all the cells (and their values) of a sheet are accessible from a VBA program. This makes possible to let cells hold variables as well as the input and output of a program. In addition, the statements, expressions, and reference mechanisms of VBA are possible to find out in an easy way; any activity can be recorded as a macro, from which the proper commands can directly be read.

Algorithm animation illustrates the behavior of an algorithm by visualizing the basic operations while the algorithm is running. Rautama *et al.* [6] explain how algorithms can be animated with Excel using data visualization and macro programming features of VBA. An algorithm is written in VBA and the animation can be observed as smoothly updating charts, which are representing the data structures of the algorithm.

Excel has several advantages as a visualization platform and a prototyping tool. Excel is versatile and easy to use, and it suits both novices and experts. An obvious advantage is its portability and low cost. Excel is available almost everywhere on the Windows and Macintosh platforms. Hence, it is easy to port and distribute Excel applications and spreadsheets.

3 Principles of Use

Flopex is an Excel application that allows the user to draw an algorithm as a flowchart and to follow the visual execution of a flowchart. At first the user constructs the flowchart by combining icons which represent control blocks and flow lines. Flopex has a collection of ready-made flowchart icons shown in Fig. 1. The icons have conventional shapes and distinct colors for easy recognition. The user copies the spreadsheet cell holding an icon from the icon palette and pastes it to a relevant location in his flowchart on a sheet.

When the flowchart has been completed, the user initializes the system by pressing the Reset button. Execution is performed by the Flopex interpreter programmed in VBA. The

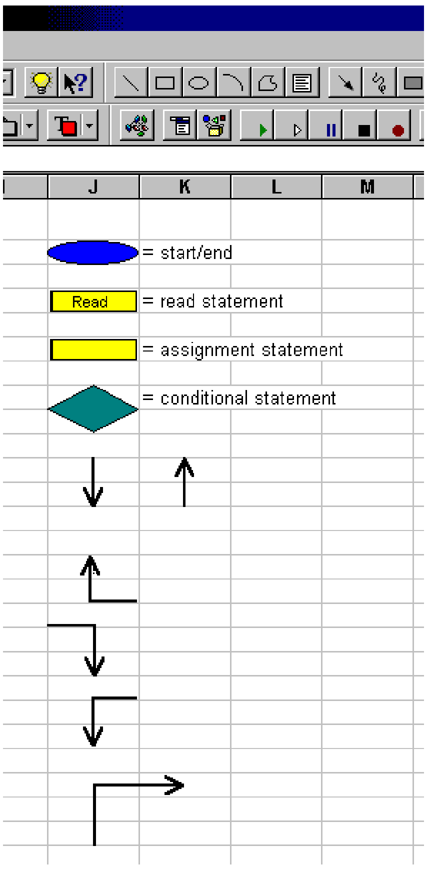

Figure 1: Icons of Flopex.

flowchart is the input of that interpreter. Interpretation starts from the start icon of the flowchart and traverses the flowchart according to the actions connected with each icon. The user can select one of two execution modes: continuous and step-by-step. The control of execution proceeds according to the direction of flow lines and the values of conditional expressions. The active cell is highlighted during execution. The continuous execution stops when the end icon is entered, an error is detected, or the Esc key is pressed. When execution has been stopped, it can be continued either in the continuous or step-by-step mode.

An assignment to a variable is done with the assignment icon. The assignment statement is written as text to the cell below the icon in the form "<variable> := <expression>". Let us first consider single variables (arrays are considered at the end of this section). The value of the right-hand side, which can be any Excel expression, is computed. If the variable of the left-hand side is new, the interpreter takes it to the variable table (columns B and C). The identifier is stored to column B and its value to column C. If the variable is already in the variable table, only its value is updated.

The read icon is interpreted as follows. A variable identifier is given in the cell below the read icon. This is called an input variable. Cells below the cell A1 in column A form the input area of Flopex. The input pointer shows the next unused cell of the input area. The value of this cell is assigned to the input variable. The variable table is updated as in the case of an assignment. The content of the input area is saved with the sheet. So one could have several versions of the same flowchart with various inputs for demonstration purposes.

The icon of conditional branch (a diamond) is entered from the top. The conditional expression is written as text to the cell below the icon in the form "if <expression>". The icon has two handles (true and false) with which outgoing flow lines can be connected. The true branch goes to the left and the false branch to the right.

The functionality of a flowchart can be checked by running it. If the control enters an error situation, the interpreter provides the user with an error message and then stops execution. It is the user's task to correct the flowchart.

An example of a flowchart is shown in Fig. 2. The example program computes the sum of squares for a list of integers. Column A holds the input area where the program reads the list. The last item is zero indicating the end of the list in our example. The input numbers are read to the variable X one at a time and the square of its value is added to the variable sum until zero is read. The user observes the result as the value of the variable sum in the cell C3.

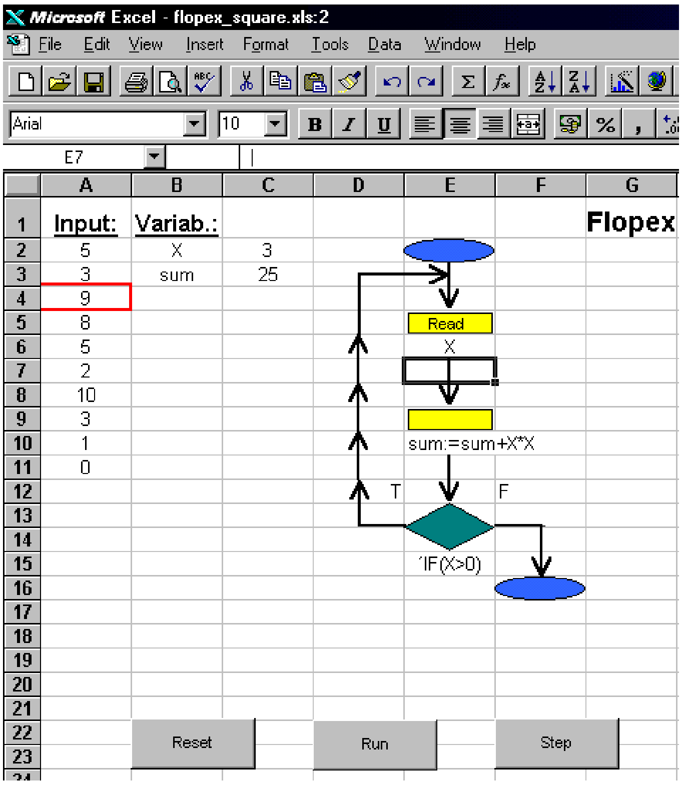

Figure 2: Example algorithm as a flowchart.

Execution is controlled with three buttons: Reset, Run, and Step. The Reset button initializes the single variables to zeros in column C. The user is responsible for the values in the input area in column A. Reset has no influence on the names of variables in column B. Reset moves both the input pointer to the beginning of the input area (to cell A2) and the cursor to the start icon.

After the initialization, the continuous execution is started with the Run button. The execution can be stopped with the break command via the Esc button. The step-by-step execution is started and advanced with the Step button.

Array variables are processed in a different way. Before execution the user should define an identifier to represent a block of cells. This is done only once with the define action of Excel. The reset operation described above has no effect on the values of array variables. The one-dimensional array reference has the form "<identifier>[<expression>]". Flopex can also handle two-dimensional arrays.

A flowchart for a sorting algorithm is show in Fig. 3. On the right there is a bar graph depicting the current situation in the array. In Section 5, we explain how the bar graph is associated with the flowchart execution.

4 Implementation

In Flopex, a flowchart drawn by the user is the input program, and the interpreter programmed in VBA executes it. The collection of icons for the flowchart construction has been drawn with the drawing tool of Excel. The identifier of an icon has been stored to a cell behind the icon. When the user copies an icon while constructing the flowchart, this identifier is copied with the icon, although the identifier is hidden from the user. The interpretation of icons is based on these identifiers. Many of the icons have additional information in the cell below the icon. There could be a condition, an input variable, or an assignment statement depending on the type of the icon.

The transition of the control from one icon to another is implemented as follows: the Flopex interpreter determines, based on the type of the icon expressed by the hidden identifier, where the control should go next. The transition associated with the line-down icon, for example, is achieved by incrementing the variable `currentRow` by one. The movement of the cursor on the screen is accomplished by activating the new cell determined by the variables `currentRow` and `currentColumn`.

The handling of single variables is based on the define mechanism of Excel. When a new variable identifier is encountered, a cell is allocated for the identifier and another cell for the value. These cells are located in the variable table in columns B and C. The identifier is defined to refer to the cell of its value. When the sheet is saved, the contents of the variable table is saved with it. Thus the names of variables will already be known in subsequent runs.

In the case of array variables, the user should define the identifier to denote the reference to a block of cells. The array references in a flowchart are transformed into sheet references by using the index function of Excel during interpretation. To name an example, the reference `A[i]` is interpreted as `index(A,i)`.

5 Characteristics

5.1 Visual Variables

In Flopex, the variables of an algorithm are represented as cells of a spreadsheet. Thus the user sees the value of a variable on the screen all the time during execution. This is an interesting feature from an educational point of view. It gives self-confidence to a novice, because the value is not hidden somewhere deep in the memory of a computer. In addition, it makes testing easier, because the intermediate values are visible during execution. Therefore no additional debugger needs to be called in the case of programming errors. It is widely known that the use of debuggers is troublesome for novices.

Excel supports multiple views for data, and the same is true for Flopex. A range of cells can be shown as a chart of any kind, and there could be even several charts of different types depicting the same range. In Flopex the user sees the numeric values as default, and in addition to this it is possible to show the values of variables in a graphic form in charts. Fig. 3 presents a sorting algorithm. The values of the array Tab are shown as numbers in column D and as a bar graph on the right hand-side of the figure.

The use of charts in this way makes simple automatic animation possible. Because Excel updates the chart after each change of any value, Flopex provides the user with a dynamic view of the current situation in the algorithm. The user needs only define the chart type and the input range before launching the interpreter of Flopex.

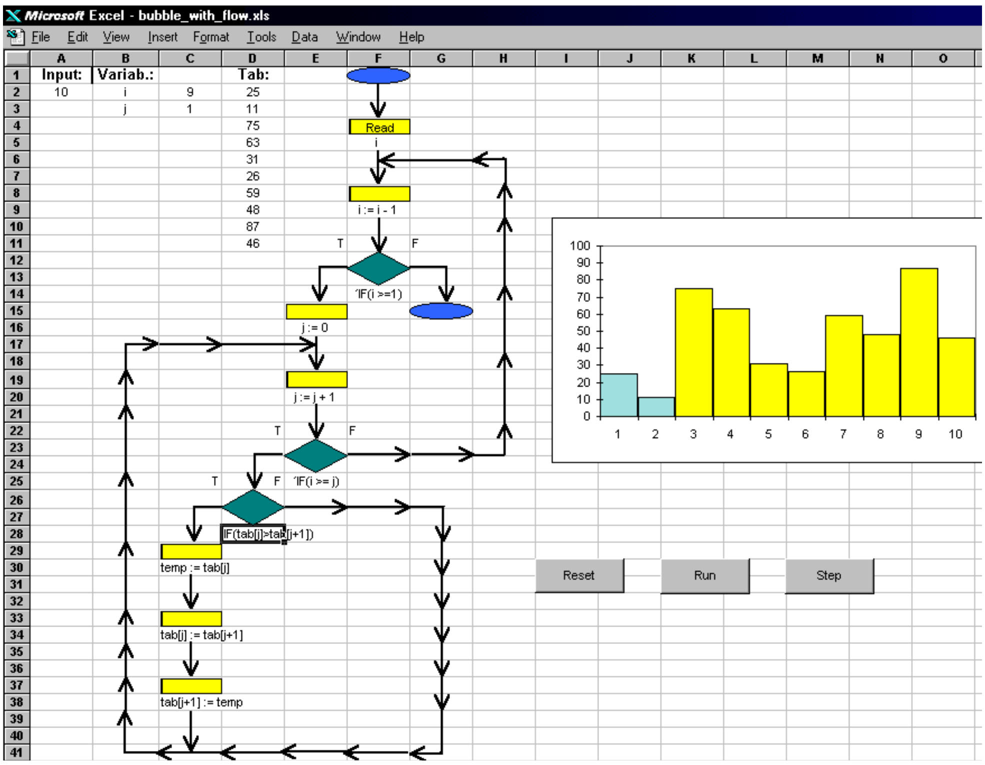

Figure 3: Sorting example.

5.2 Adaptability

With Flopex, the user can construct and test flowcharts for algorithms of many types. Because the right-hand side of an assignment can be any Excel expression, the user has full access to ready-made functions and graphic routines of Excel. This opens possibilities to apply Flopex in advanced and unusual ways, as for example, taking advantage of the searching functions of Excel to find a certain record in a database.

Flopex inherits the openness of Excel: the application is easy to modify. Because the interpreter of Flopex is written in VBA and it runs with the VBA interpreter of Excel, the code of the interpreter is available for any user. For example, to add a new flowchart icon, one should design the appearance and write a short piece of code to interpret it. In practice, the code can be modified from the code of a related icon.

The current version of Flopex only supports numeric variables, but it is straight-forward to implement other basic types like char and boolean.

Flopex is a portable application, because Excel is widely available. The lightweight implementation encourages to make modifications to the system. The lightness is due to the expressive power of Excel.

6 Concluding Remarks

Flopex is suitable for students at many levels. Beginners can learn and practise programming, and advanced students could extend or modify the system. A teacher can use Flopex as a demonstration tool, e.g. for visualizing flowchart execution in teaching elementary programming. In addition, Flopex can be applied to the visualization of non-CS processes. Flopex can even be used to illustrate complicated calculations in a spreadsheet or to program graphics.

Besides these basic applications, Flopex has potential for new uses. The same is true for spreadsheet programs. We challenge the reader to explore the capabilities of modern spreadsheet programs in order to find new applications. Integration of a programming environment with spreadsheet graphics has implicit expressive power. We claim that all possible advanced uses have not yet been discovered.

Acknowledgements. The authors thank Janne Markkanen, Erkki Sutinen and Tommi Teräsvirta for fruitful discussions. The work was financially supported by the Academy of Finland.

References

[1] Calloni, B. A. and Bagert, D. 1997. Iconic Programming Proves Effective for Teaching the First Year Programming Sequence. In *Proc. SIGCSE'97, Technical Symposium on Computer Science Education, SIGCSE Bulletin, Volume 29*, 1, 262–266.

[2] Dybdahl, A., Sutinen, E., and Tarhio, J. 1998. On Animation Features of Excel. In *Proc. ITiCSE '98, Integrating Technology into Computer Science Education*. ACM, 77–80.

[3] Dybdahl, A., Sutinen, E., and Tarhio, J. 1998. Excel as a Teaching Tool. In *Proc. LeTTeT '98, Learning Technology and Telematics in Education and Training*. Pori, Finland, 35–39.

[4] Neuwirth, E. 2000. Spreadsheets, Mathematics, Science, and Statistics Education [seen on Feb. 4, 2000]. <URL: http://sunsite.univie.ac.at/Spreadsite/>.

[5] Ragsdale, C. 1995. *Spreadsheet Modeling and Decision Analysis: a Practical Introduction to Management Science*. Course Technology, Cambridge, MA.

[6] Rautama, E., Sutinen, E., and Tarhio, J. 1997. Excel as an Algorithm Animation Environment. In *Proc. ITiCSE '97, Integrating Technology into Computer Science Education*. ACM, 24–26.

[7] Scanlan, D. A. 1989. Structured Flowcharts Outperform Pseudocode: An Experimental Comparison. *IEEE Software 6*, 128–36.

[8] Ziegler, U. and Crews, T. 1999. An Integrated Program Development Tool for Teaching and Learning how to Program. *SIGCSEB: SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education) 31.*