

# Developing the algorithmic skills through word processing and handling spreadsheets

CSERNOCH Mária<sup>1</sup>, BUJDOSÓ Gyöngyi<sup>2</sup>

<sup>1</sup>Department of Information Technology and <sup>2</sup>Department of Computer and Library Information Science

University of Debrecen, Faculty of Science and Technology

Egyetem tér 1., 4010 Debrecen, Hungary,

E-Mail: [csernoch.maria@inf.unideb.hu](mailto:csernoch.maria@inf.unideb.hu), [bujdosog.yongyi@inf.unideb.hu](mailto:bujdosog.yongyi@inf.unideb.hu)

**Abstract** – Compared to the high number of computer users the proportion of those students who are willing to write programs has dropped seriously. Even those whose curriculum includes programming usually start late and the chance to change their attitude towards programming lessens as they are getting older. A possible solution would be to sneak programming into the primary and secondary education by hiding it behind the scenes of software they use. Students are usually proud of their knowledge of word processing and handling spreadsheets and they think these programs are easy to use. We can thus use these programs to teach algorithms instead of using traditional environments. In this article, through the analysis of a problem, we give examples how we can switch from programming languages to applications and still teach some basic algorithmic skills.

**Keywords:** teaching programming, forming algorithms, spreadsheets, applications.

## I. INTRODUCTION

We all experience that with the increasing number of computers the proportion of those who use programming languages as compared to those who are using application software has dropped seriously. It has become a challenging task to convince average students that they should learn programming. In this novel environment it is the teachers' responsibility to bring back programming and more importantly developing algorithmic skills in Informatics classes [2]. A well-chosen task by itself can be a good starting point, but beyond the task, the different methods, which can be used to solve the problem, would have more influence on the students.

When teaching programming we solve a problem in several different ways and discuss the advantages and disadvantages of each solution found. The same can be done to tasks which are wrapped in word processing or handling spreadsheets. Students believe that they are creating a text, a table, and nothing else, but there is a lot more behind the scenes.

Here, we present a task from one of the Competitions of Applied Informatics of Hungary [1, 3], and through this task a couple of different possibilities to solve it. To

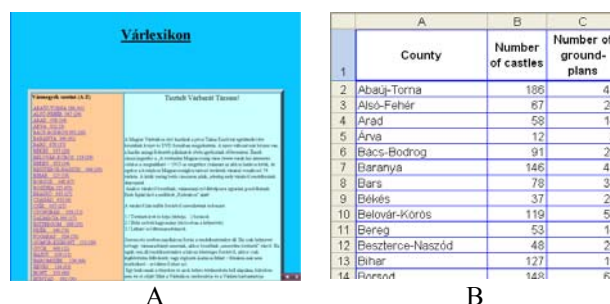
break the task into meaningful smaller parts gives opportunities to get acquainted with basic programming algorithms.

In this project Microsoft Word and Excel are used to carry out this selected task, but other software such as Open Office, would be equally acceptable.

## II. RESULTS

The task we will use to demonstrate how word processors and spreadsheets can be used for teaching algorithmic thinking is presented in Fig. 1.

### A. Converting text from a web page into a spreadsheet



	A	B	C
1	County	Number of castles	Number of ground-plans
2	Abaúj-Torna	186	41
3	Alsó-Fehér	67	24
4	Arad	58	14
5	Arva	12	3
6	Bács-Bodrog	91	26
7	Baranya	146	41
8	Bors	78	37
9	Békés	37	20
10	Belovár-Körös	119	59
11	Bereg	53	14
12	Beszterce-Naszód	48	29
13	Bihar	127	19
14	Hajdú-Bihar	148	67

Fig. 1. The first task to be solved. A table of three columns should be created with a spreadsheet, as drawn in B, enlarged in Fig. 3, from the left column of the given web page (A, enlarged in Fig. 2). The first column of the web page consists of three data: the name of a county, the number of castles in this county, and the number of castles with ground-plan (in this order). These data should be converted into the first three columns of the spreadsheet.

Being familiar with the content of the first column of the web page might help the conversion process. To take a closer look at these data we can open the file in a browser (Fig. 1A), in MS Word (Fig. 5), in Excel (data not shown), or the source of the web page in a text editor (Fig. 4). Opening the source or the file in a word processor or a text editor reveals that the text is loaded with both normal and Non-breaking Spaces in arbitrary order and places (Fig. 4 and 5). These Space characters should be regularized to be able to convert the text into the desired three columns of a spreadsheet.



Fig. 2. The original web page, whose first column (peach background on the left) should be converted into a spreadsheet.

	A	B	C
	County	Number of castles	Number of ground-plans
1			
2	Abaúj-Torna	186	41
3	Alsó-Fehér	67	24
4	Arad	58	14
5	Árva	12	3
6	Bács-Bodrog	91	26
7	Baranya	146	41
8	Bars	78	37
9	Békés	37	20
10	Belovár-Körös	119	59
11	Bereg	53	14
12	Beszterce-Naszód	48	29
13	Bihar	127	19

Fig. 3. A table with three columns should be generated from the data of the first column in Fig. 2.



Fig. 4. The source of the web page presented in Fig. 2 when opened in a text editor. The text is loaded with normal and Non-breaking Spaces. Arrow points to the extra Non-breaking Space following county Bosnia, inclosed in the Anchor tag (<A></A>).

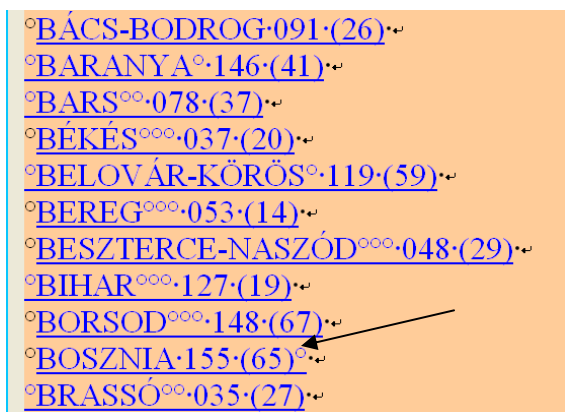


Fig. 5. The printing and non-printing characters (Spaces, Non-breaking Spaces, Manual Line Breaks) of the first column of the web page presented in Fig. 2 when opened in MS Word. Arrow points to the extra Non-breaking Space following county Bosnia.

#### A.1. Solving the task in Excel

We can directly open the web page in Excel. If we do this, we first have to delete the unnecessary rows above the name of the counties and the unnecessary columns right of them. Doing this gives us a single column with three data in each cell. If we save the file as a text file and then reopen it in Excel while using Space as separator character we generate a three-columned structure.

At first sight the conversion worked. In the first column we have the name of the counties in capital letters, in the second the number the castles, and finally, the number of the ground-plans as negative numbers, due to the parentheses around these numbers on the web page. However, the conversion does not work completely in the third column in the row of 'BOSZNIA', where the number remains in the original format, with the parenthesis. This is due to the extra Non-breaking Space following the closing parenthesis

(Fig. 4, 5), which turns the negative value into a string. To correct this problem the extra Non-breaking Space should be deleted.

Now we have to set up the algorithm to bring the three columns into the correct format (Fig. 3).

- 1) Compute the absolute value of column C.
- 2) Delete the leading Spaces of column A.
- 3) Delete the follow up Spaces of column A.
- 4) Change the case of column A to Title case.
- 5) Delete the substitute columns from the table.

The solutions for these steps are detailed below.

#### A.1.1. *Computing the absolute value of column C*

After we have negative numbers in each row, the absolute value of the numbers of column C should be calculated and put into column D.

D2: =ABS(C2)

#### A.1.2. *Deleting the leading Spaces of column A*

To delete the leading Space from each cell, we have to crop the string from the right of the cell to characters numbering length - 1.

E2: =RIGHT(A2,LEN(A2)-1)

#### A.1.3. *Deleting the follow up Spaces of column A*

In the next step we have to crop the string of column E to the meaningful part by deleting the Spaces, if there is any, from the end of the cell. To do this, first we have to find out if there is any extra Space or not. If the cell is burdened with Space(s) we have to find the position of the first Space.

Both questions can be answered with the SEARCH() function. If we have extra Space(s) the function returns the position of the first Space. If there is no such Space, the function comes back with a #VALUE!. To handle both cases an enclosing IF() is needed. If SEARCH() returns with an error the whole content of cell E2 should be copied to F2. If SEARCH() returns with an integer the text should be truncated with the LEFT() function until the position found.

F2: =IF(ISERROR(SEARCH(" ",E2)),  
E2,LEFT(E2,SEARCH(" ",E2)-1))

There is however one more detail which should be taken care of, namely, the first actual parameter of function SEARCH(). Since the extra characters are Non-breaking Spaces the first parameter of the function must be a Non-breaking Space, which should be copied into the function from column A. Note, that we can check the result by copying the values of column F to G by **Paste Special**.

#### A.1.4. *Changing the case of column A to Title case*

The last step of this process is to change the case of the names of the counties from **Capital** to **Title case**. Here, again, an Excel function, PROPER(), can be used to finish the conversion into column H. Both F and G can be used as the actual parameter of PROPER.

H2: =PROPER(F2)

#### A.1.5. *Deleting the substitute columns from the table*

Finally, the substitute columns should be deleted and the three columns should be ordered according to the sample (Fig. 3). Be aware that a simple deleting does not work. If a column which is referred to is deleted the function loses its source and consequently its values. Copying the values with **Paste Special** (see in Chapter A.1.3) solves the problem.

To complete this assignment was like writing a program. First, we had to find the algorithm, then the functions which are available in this environment to fulfill the task, call these functions with the actual parameter(s), and finally create embedded functions if it was necessary. Students not necessarily realize that they are programming, because the surface of Excel does not offer the familiar programming environment. Nevertheless, they do. We believe, if they move more freely in the Excel environment, let them do so. It is the teachers' responsibility to guide them from behind the scenes towards programming and forming algorithms.

#### A.2. *Solving the task by opening the file in a browser or in Word*

Other different algorithms can be assigned to this task if we complete it using a word processor such as MS Word. In this case we should convert the first column of the web page (Fig. 2) to a table with three columns in the word processor through a sequence of **Replace** commands.

Actually, we give two methods, which means two independent algorithms. To unify the two different kinds of Spaces in the first algorithm they are converted into Non-breaking Spaces, while in the second they are converted into normal Spaces.

If we open the web page in either a browser or in a word processor we can copy the column in question into a new Word document. The difference between the two algorithms lies in the type of Paste we use. By using the normal Paste command both the normal and the Non-breaking Spaces are inherited together with the line closing Manual Line Break (Shift+Enter) characters and the hyperlinks. By using the **Paste Special** command and selecting **Unformatted Unicode Text** from the list of **Paste Special** dialog box we come to a text with normal Spaces only, with line-closing End of paragraph (Enter) characters.

### A.3. *First algorithm, using normal Paste command*

The text is opened in a browser or in Word and the text in question is copied to a new document using normal **Paste**. In this new document first the normal Spaces should be converted to Non-breaking Spaces to unify them. The two fields of the **Replace** command should be filled in as follows.

**Find what:** Space  
**Replace with:** ^s

Both **Find what** and **Replace with** fields can be filled in by typing the codes of the corresponding non-printing characters or selecting them from the list offered by **More/Special**. If the two fields are filled in we can choose **Replace All** to complete the replacement in one step.

Change the Manual Line Break characters to Paragraph Mark characters.

**Find what:** ^l  
**Replace with:** ^p

Change the 'Non-breaking Space-Enter' combinations to Enter.

**Find what:** ^s^p  
**Replace with:** ^p

Change the multiple Non-breaking Spaces to single Non-breaking Spaces. Repeat the **Replace all** command until the number of replacements is down to zero.

**Find what:** ^s^s  
**Replace with:** ^s

Delete the opening parenthesis.

**Find what:** (  
**Replace with:**

Delete the closing parenthesis.

**Find what:** )  
**Replace with:**

Change the Non-breaking Spaces to Tabulator characters

**Find what:** ^s  
**Replace with:** ^t

With this final replacement, we have a table of four columns. The unnecessary leading column can easily be deleted in Excel. Finally, we have to change the case of the names of the counties to **Title case**.

Now we have at least two choices to import the file into Excel. We can copy the table to Excel with **Paste Special** to get rid of the hyperlinks, or we can save it as a text file and open it in Excel.

### A.4. *Second algorithm, using Paste Special command*

**Paste Special** is like a stepchild. It does not matter how good it is only a small proportion of users prefer it against normal Paste. In the following solution we present how much easier it is to solve this problem with **Paste Special** than with normal **Paste**.

Copying the text of the web page with the **Paste Special** command into a new document results in a text

which carries only normal Space characters, Enters at the end of the lines without hyperlinks.

Similarly to the previous algorithm (A.4) a sequence of replacements should be carried out.

Change the multiple Spaces to single Spaces. Repeat the **Replace all** command until the number of replacements is down to zero.

**Find what:** SpaceSpace  
**Replace with:** Space

Change the 'Space-Enter' combinations to Enter.

**Find what:** Space^p  
**Replace with:** ^p

Change the Spaces to tabulators

**Find what:** Space  
**Replace with:** ^t

Delete the opening parenthesis.

**Find what:** (  
**Replace with:**

Delete the closing parenthesis.

**Find what:** )  
**Replace with:**

After creating the four-columned table we have to change the case of the characters and then copy the text into Excel with **Paste**, with **Paste Special**, or we can save it as a text file for further use.

Either of the detailed methods is good for guiding the students to find the algorithms for the correct order of the replacements. The students should find out which steps had to be completed previous to the another. If they are able to see clearly that only one Space character should be left between the two columns and this replacement should be prior to the 'Space-Tab' replacement then they get the hint of how to form an algorithm.

## B. *Condition based coloring of a spreadsheet*

### B.1. *The algorithm of the problem*

The tool in Excel to complete the assignment is Conditional Formatting. However, previous to Conditional Formatting we have to create a column with substitute values which indicates when a change in the first character of the names occurs.

- 1) Set an initial value for the first cell of the substitute column.
- 2) Cut the first character of the name.
- 3) Create a condition to distinguish the first characters.
- 4) Handle the case when there is no change in the first characters.
- 5) Handle the case when there is a change in the first character.
- 6) Set conditional formatting.

Two opportunities are available. We can either use any two distinguishable characters (two letters of the alphabet, two words longer than one character, two

number, etc.), or the two Boolean values for the substitute values.

	A	B	C
1	County	Number of castles	Number of ground-plans
2	Abaúj-Torna	186	41
3	Alsó-Fehér	67	24
4	Arad	58	14
5	Árva	12	3
6	Bács-Bodrog	91	26
7	Baranya	146	41
8	Bars	78	37
9	Békés	37	20
10	Belovár-Kőrös	119	59
11	Bereg	53	14
12	Beszterce-Naszód	48	29
13	Bihar	127	19
14	Borsod	148	67
15	Bosznia	155	65
16	Brassó	35	27
17	Csanád	10	4
18	Csik	57	27
19	Csongrád	30	11
20	Dalmácia	66	17
21	Esztergom	88	33

Fig. 6. The second task to be solved. The names of the counties are in alphabetical order in column A. The background color of the cells should alternate between yellow and blue depending on the first character of the name, that is the background color of the cells with names starting with the same character should be colored with the same color. Note that the letters A and Á are different in the Hungarian alphabet.

**B.2. First algorithm using two characters as substitute values**

Let us start with the longer but simpler method. For a beginner the method where two easily distinguishable values alternate is easier to follow. To distinguish the two states we use the one-character-long strings “Y” (for yellow) and “B” (for blue, Fig. 7).

In the steps detailed below the embedded functions are built up from the most enclosed towards outward, following Steps 1–5 of B.1.

**B.2.1. Setting an initial value for the first cell**

To start the process the first substitute cell (D2) should be set to either “Y” or “B”. We start with “Y”.

**B.2.2. Cutting the first character of the name**

In the following cells, starting with D3, we have to compare the first character of cell A2 to the first character of cell A3. The first character of a string can be excised with the function LEFT(). Since only the first character is needed, the second parameter of function LEFT() can be omitted.

D2: =LEFT(A2)

**B.2.3. Creating a condition to separate the first characters**

The comparison can be carried out with function IF(). In the condition of IF() the results of the two LEFT() functions are compared.

D3: =IF(LEFT(A2)=LEFT(A3),,,)

**B.2.4. Handling the case when there is no change in the first characters**

If the two characters are identical then the character in D2 should go into D3.

D3: =IF(LEFT(A2)=LEFT(A3),D2,)

	A	B	C	D
1	County	Number of castles	Number of ground-plans	Y/B
2	Abaúj-Torna	186	41	Y
3	Alsó-Fehér	67	24	Y
4	Arad	58	14	Y
5	Árva	12	3	B
6	Bács-Bodrog	91	26	Y
7	Baranya	146	41	Y

Fig. 7. Substitute values in column D when characters are used.

**B.2.5. Handling the case when there is change in the first characters**

If the two characters are different we have to check the content of D2. If it was a “Y” we have to set D3 to “B”, otherwise to “Y”.

D3: =IF(LEFT(A2)=LEFT(A3),D2,IF(D2="Y","B","Y"))

**B.2.6. Setting conditional formatting**

To color the cells we have to add the conditions to the panels of Conditional Formatting (Fig. 8).

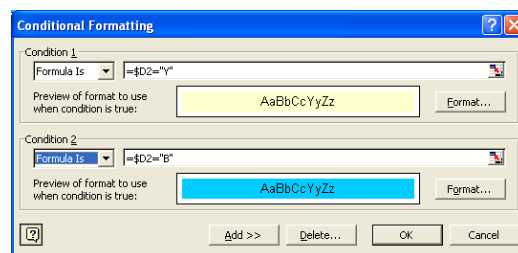


Fig. 8. The filled in panels of Conditional Formatting when ‘Y’ and ‘B’ characters are used as substitute values.

**B.3. Second algorithm using two Boolean values for the substitute values**

If we use “TRUE” or “FALSE” as substitute values (Fig. 9) we can omit the embedded IF function. However, we still have to be sure to change from one Boolean value to the other when the first character of the names changes. That is, if the two starting characters are

the same we have to set E3 to the value of E2, if they are not, we have to set E3 to the opposite of E2. The opposite of a Boolean value can be generated with the function NOT().

E3: =IF(LEFT(A2)=LEFT(A3),E2,NOT(E2))

	A	B	C	D	E
1	County	Number of castles	Number of ground-plans	Y/B	TRUE/FALSE
2	Abaúj-Torna	186	41	Y	TRUE
3	Alsó-Fehér	67	24	Y	TRUE
4	Arad	58	14	Y	TRUE
5	Árva	12	3	B	FALSE
6	Bács-Bodrog	91	26	Y	TRUE
7	Baranya	146	41	Y	TRUE

Fig. 9. Boolean substitute values in column E. The substitute values of 'Y' and 'B' are left in column D to make the comparison of the substitute values more obvious.

Using the Boolean values for the substitute values the conditions of the Conditional Formatting is somewhat shorter than they are in the previous algorithm (Fig. 10).

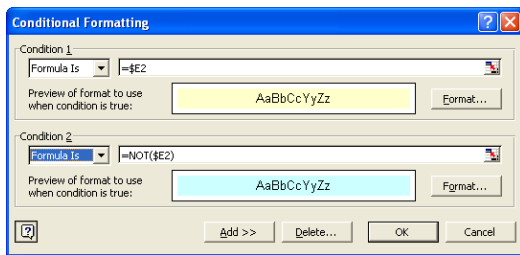


Fig. 10. The filled in panels of Conditional Formatting when Boolean values are used as substitute values.

In this coloring task we met the classical programming problem how we can distinguish two

states. The algorithms presented here are exactly the same we would use in a source code of a program.

### III. CONCLUSIONS

In the problems given in this article we demonstrate that teaching basic algorithms does not necessarily require a classical programming environment. We therefore believe, if the students feel more comfortable in an application let them use it. The focus should be on the task not on the programming environment. With this method even those students who do not want to be a professional in Informatics can get the hint of forming algorithms. The age has an important role in this process, since to get acquainted with programming as an adult is much harder than as a child. If the students first encounter such problems at an early age later on they would absorb it much easier.

The specific assignments presented in the article and the chosen applications, however, are just one of many possible examples. Several other opportunities are open to find similar tasks and wrap them in the environment of familiar and popular applications.

### REFERENCES

- [1] Competitions of Applied Informatics of Hungary [http://tehetseg.inf.elte.hu/nemesa/2008/nemes\\_adat1.zip](http://tehetseg.inf.elte.hu/nemesa/2008/nemes_adat1.zip)
- [2] L. Raymond *After the gold rush: toward sustainable scholarship in computing*. Conferences in Research and Practice in Information Technology Series; Vol. 315 Proceedings of the tenth conference on Australasian computing education - Volume 78, 2008.
- [3] L. Zsakó *Combinatorics – Competition – Excel*. Teaching Mathematics and Computer Science.4 (2006) 2. pp. 427–435, 2006.