TALLINN UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

Department of Computer Systems

Andreas Hofmann – 177873MV

Homework 2

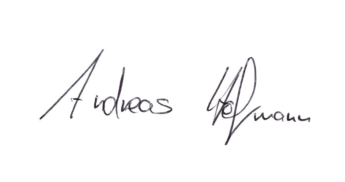
Programming C - IAX0583

**Supervisor(s):**  Marina Brink

Sergei Kostin

# Declaration of authorship

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. All works and major viewpoints of the other authors, data from other sources of literature and elsewhere used for writing this paper have been referenced.



Andreas Hofmann, 27.11.2017, Tallinn

Contents

[Declaration of authorship 2](#_Toc499590747)

[Variant 5: Cipher Keys 4](#_Toc499590748)

[Brief overview of the task 5](#_Toc499590749)

[ArgoUML – Activity Diagram 6](#_Toc499590750)

[Program listings with corresponding comments 7](#_Toc499590751)

[Additional verbal description of my solution 12](#_Toc499590752)

[Snapshots 14](#_Toc499590753)

[Verification of results by using SciTE 18](#_Toc499590754)

# Variant 5: Cipher Keys

Variant 5 is called “Cipher Keys”. The variant is called so because one must calculate the cipher keys by summing up the digits of the amount of gold bars in one cell. Cipher keys are part of cryptography and is an algorithm for performing encryption or decryption by using well-defined steps that follow a procedure.

The program is coded in the language C and is using an array with two columns to calculate and display the end result. Furthermore, there have to be at least four functions without the main function. Moreover, in the later part of the documentation an ArgoUML chart will be shown, some snapshot with description of the program and two ScieTE texts will be shown.

# Brief overview of the task

As already mentioned above, in this task the first step is that the user is prompt to enter a value ‘N’ which is the number of cells the bank has. Furthermore, the program must check if the value entered by the user is in between the boundaries [1…15]. This is the first function.

The second function is the filling of the array. Here the program asks for every cell again and does not use random numbers. Furthermore, we assume here the user enters only positive numbers, since negative numbers would implement that more gold bars have been taken out that ever have been in the cell which would make absolutely no sense.

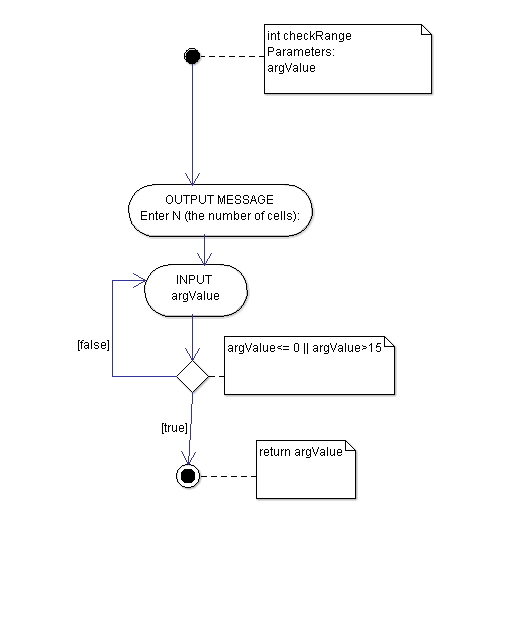
Third function is the process of filling the second column of the array with the cipher keys. In this case it is rather simple by summing up the single digits which gives us then the cipher key.

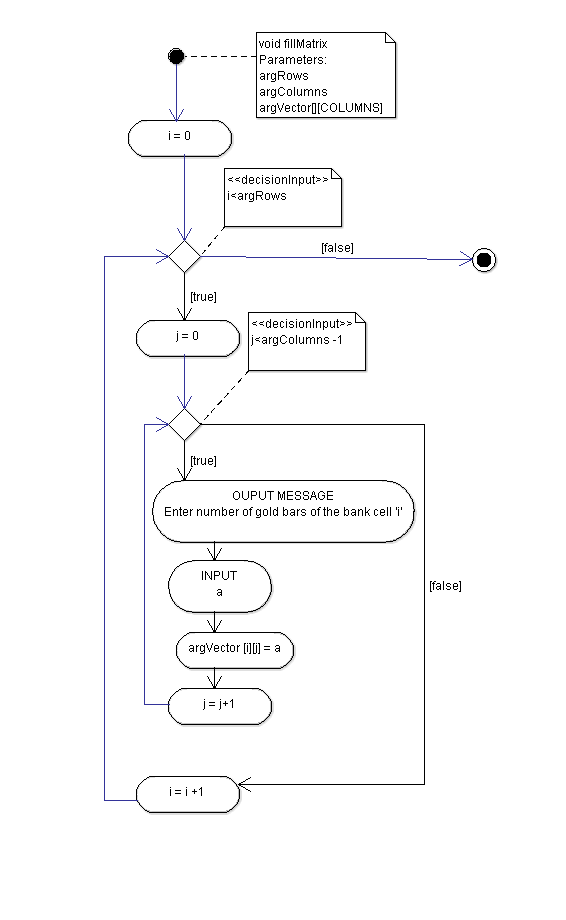
During the fourth function the matrix is displayed. The first column is independent from the array and will just show the number of the bank cell. Column two are the number of gold bars in this cell and column three are the cipher keys. The second and third column are both part of the array.

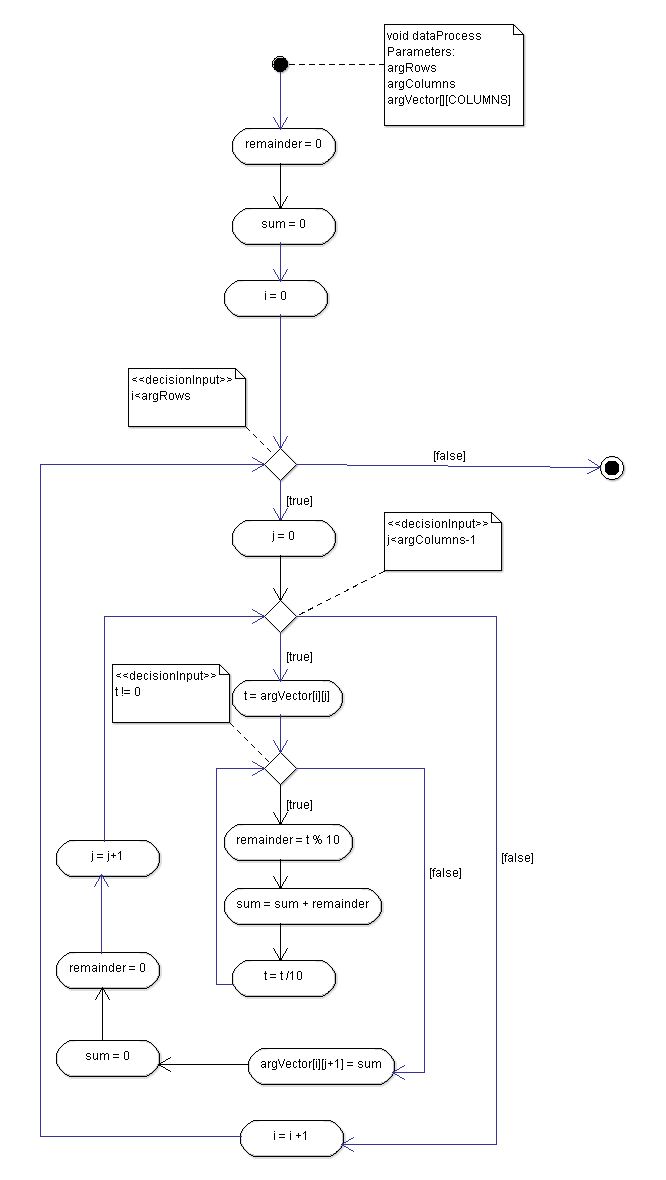
The fifth function displays the number of the cell which contains the most amount of gold bars. Especially this function is more complicated compared to the others since it involves the execution of a program that finds the highest number of gold bars which is in the array and then the corresponding number of bank cell which is more difficult since these are independent from each other and not in the same array.

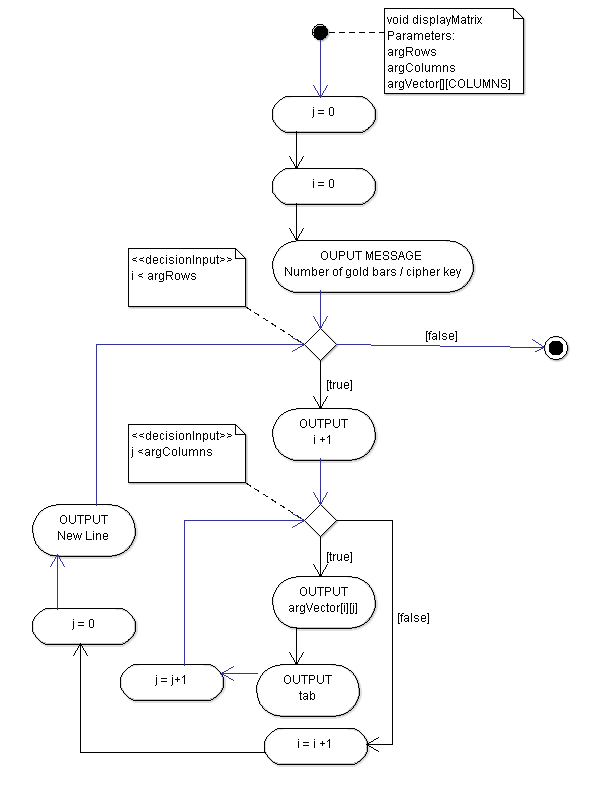
# ArgoUML – Activity Diagram

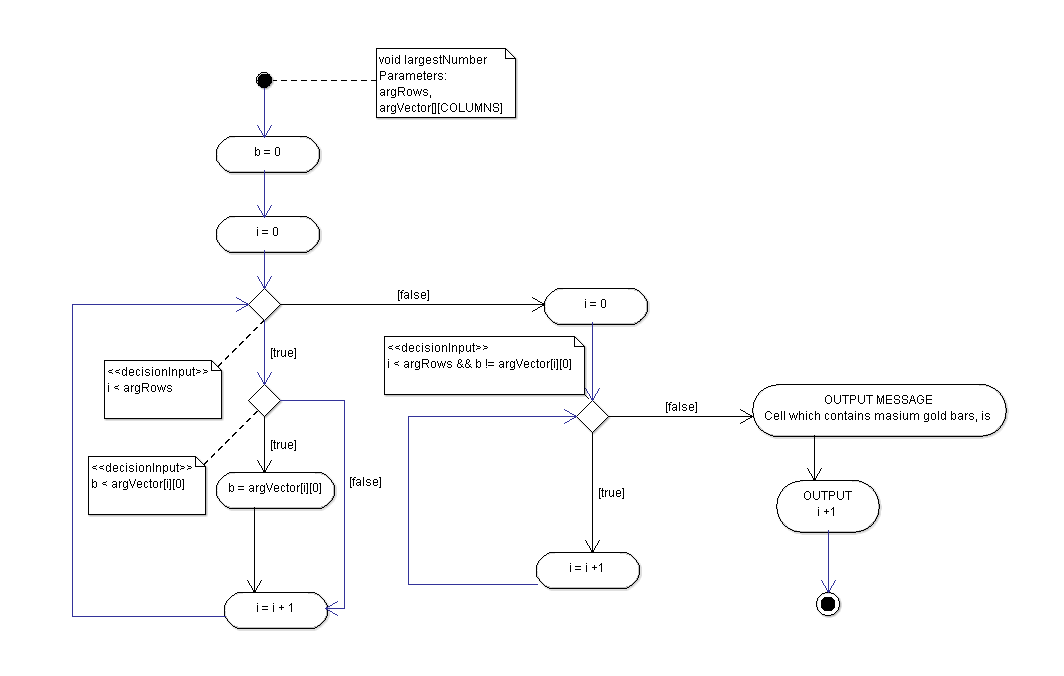
Function: int main (void)

Function: checkRange

Function: fillMatrix

Function dataProcess

Function: displayMatrix

Function: largestNumber

# Program listings with corresponding comments

/\*

\* File: HW\_1\_Andreas\_Hofmann\_177873IASM\_5

\* Author: Andreas Simon Hofmann

\* Date: 22.11.2017

\* Update: 24.11.2017

\* Update: 25.11.2017

\* Update: 26.11.2017

\* Update: 27.11.2017

\* Description: function to calculate the cipher keys of bank cells and display the cell that contains the most gold bars

\*/

#include <stdio.h>

#define COLUMNS 2

#define CELLS 15

int checkRange (int argValue);

void fillMatrix (int argRows, int argColumns, int argVector[][COLUMNS]);

void dataProcess(int argRows, int argColumns, int argVector[][COLUMNS]);

void displayMatrix (int argRows, int argColumns, int argVector[][COLUMNS]);

void largestNumber(int argRows, int argVector[][COLUMNS]);

int main (void)

{

int Value, cells;

int matrix [CELLS][COLUMNS];

Value = 0;

cells = checkRange(Value);

fillMatrix (cells, COLUMNS, matrix);

dataProcess (cells, COLUMNS, matrix);

printf("\n");

displayMatrix (cells, COLUMNS, matrix);

printf("\n");

largestNumber(cells, matrix);  
return 0;

}

/\* checkRange - a function to ensure that the entered number is [1...15] as stated in task

Parameters:  
argValue: Number entered by the user which is in this case referred as number of bank cells  
Return Value: checked number of bank cells

\*/

int checkRange (int argValue) {

do

{

printf("Enter N (the number of bank cells): \t");

scanf("%d", &argValue);

} while (argValue <=0 || argValue>15);

return argValue;

}

/\* fillMatrix - a function to fill the first Column of the vector matrix with number of gold bars in cell

\*/

void fillMatrix (int argRows, int argColumns, int argVector[][COLUMNS]) {

int i, j, a;

for(i=0; i<argRows; i++)

{

for (j=0; j<argColumns-1; j++)

{

printf("Enter the number of gold bars of the bank cell %d:\t", i+1);

scanf("%d", &a);

argVector[i][j]= a;

}

}

return;

}

/\*dataProcess – a function to calculate the Cipher keys which is the sum of the digits of the gold bars in the cell

Parameters:

argRows: number of rows

argColumns: number of columns

argVector: matrix where the number of gold bars is stored which will be summed up which will be then the cipher keys

Return value: none

\*/

void dataProcess (int argRows, int argColumns, int argVector[][COLUMNS]) {

int i, j, t, sum, remainder;

remainder = sum = 0;

for (i=0; i<argRows; i++)

{

for (j = 0; j<argColumns-1; j++)

{

t = argVector[i][j];

while (t != 0)

{

remainder = t % 10;

sum = sum + remainder;

t = t / 10;

}

argVector[i][j+1] = sum;

sum = 0;

remainder = 0;

}

}

return;

}

/\*displayMatrix – a function to display elements of the vector with first column number of gold bars and second column cipher keys

Parameters:

argRows: number of rows

argColumns: number of columns

argVector: matrix with the number of gold bars and cipher keys

Return value: none

\*/

void displayMatrix (int argRows, int argColumns, int argVector[][COLUMNS]) {

int i, j;

j=0;

i=0;

printf("The number of gold bars/cipher key:\n\n");

while (i<argRows) {

printf("%d. ", i+1);

while (j<argColumns)

{

printf("%d\t", argVector[i][j]);

j++;

}

i++; j=0;

printf("\n");

}

}

/\*largestNumber – a function to find and display the bank cell where the most gold bars are allocated

Parameters:

argRows: number of rows

argVector: matrix with the number of gold bars and cipher keys

Return value: none

\*/

void largestNumber(int argRows, int argVector[][COLUMNS]) {

/\*start searching largest number\*/

int i, b;

b = 0;

for(i = 0; i < argRows; ++i)

{

if(b < argVector[i][0])

b = argVector[i][0];

}

/\*start comparing numbers\*/

i = 0;

while (i < argRows && b != argVector[i][0])

{

i++;

}

//\*when number is found\*/

printf("The cell, which contains the maximum of gold bars, is %d", i + 1);

return;

}

# Additional verbal description of my solution

In my solution I tried to use as much functions as possible since in my opinion this makes the process of running and understanding the program easier.

During the process of coding the first function no real difficulties occurred while there was a problem later in function three. The problem was that I had declared the array with the rows ‘cells’ before I declared the value of the variable ‘cells’ which gave me the error that there was no memory for my calculations in the data process function.

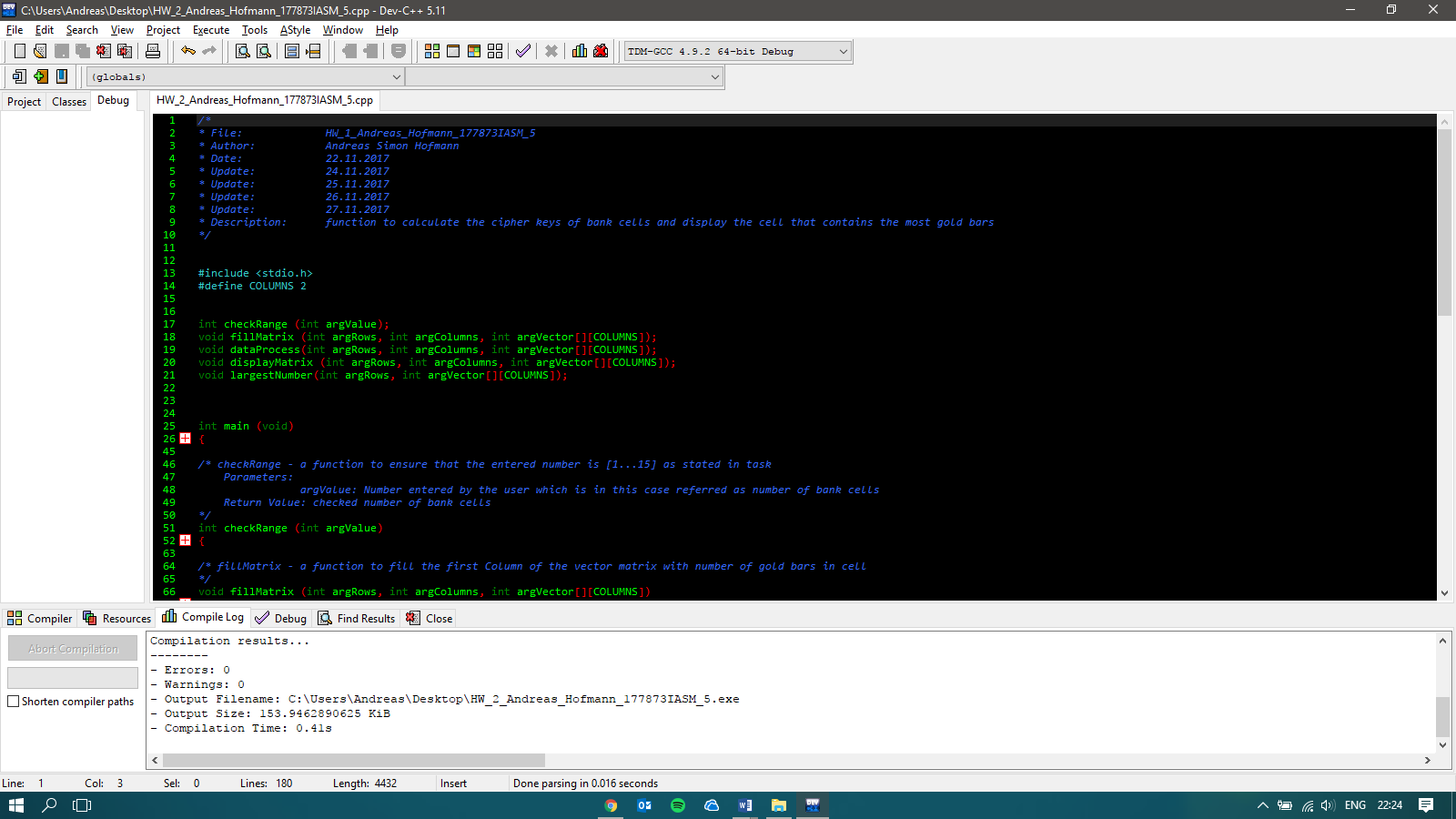
Coding the second function was, compared to the other functions, the easiest one since this is simply two for loops while the first one increases the row and the second one is prompting the user to enter the number of gold bars and thereby only filling the first column of the array.

The third function was compared to the other ones the most difficult one. First of all, it was difficult to identify what loop I should use. However, after I figured out the correct loop I had the problem that no memory was allocated to this array since it displayed me the error “Program received signal SIGSEGV, Segmentation fault.’. After coding, executing and trying different method I came to the right solution put the function of checking the boundaries of the cells above the declaration of the matrix since this helped me and was the way to the correct solution. In this function I used two for loops in order to calculate the cipher keys. The first one has the purpose to increase the row and the second one has the purpose to calculate the cipher keys and allocate it to the second column of the array. To calculate the cipher keys, I used the method of the remainder division since in my opinion it is the most suitable and easiest one for this purpose.

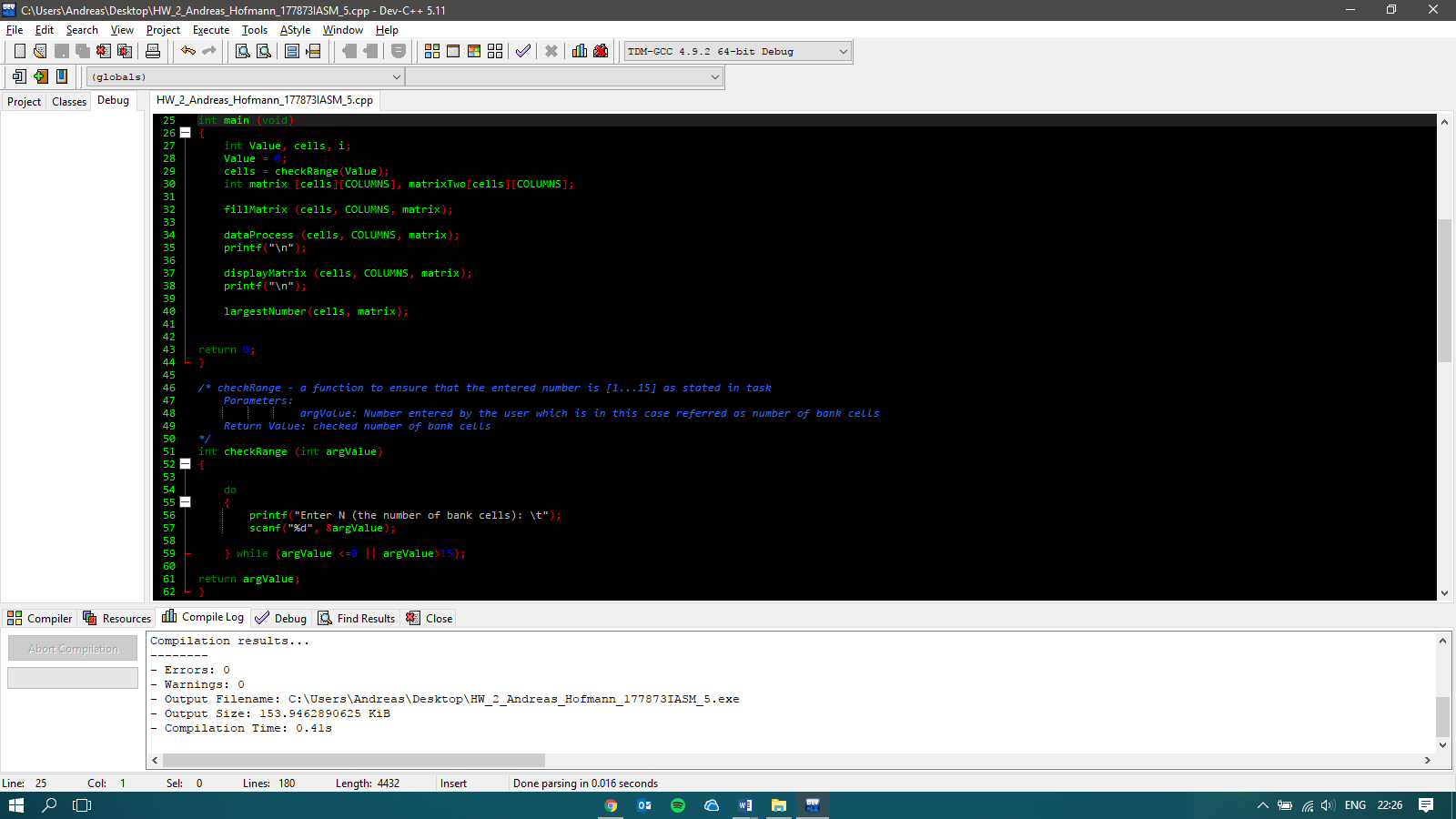
Displaying the matrix was in fact not a big problem. In other words, by using two while loops and two print functions the whole function is coded.

The last function which serves the purpose of finding the largest amount of gold bars in the array and then displaying the bank cell was right after the third function the most difficult ones. The difficult part is that the bank cell number and the array are two separate strings which made it difficult. However, I used a for loop that serves the purpose to find the largest number. I hereby used the method by comparing the single numbers of the array. If the number is bigger than the previous one it is stored in the control variable ‘b’, if not the array increases, but only the first column since we are only interested in the amount of the gold bars and not the cipher keys. Then I used a while loop that first of all checked if the control variable I is in the length of the array (between 1 and 15 depends on the value the user inputs) and secondly comparing the found value in the previous for loop with every singly number of the array while increasing the control variable ‘i’ so that the bank cell can be found.

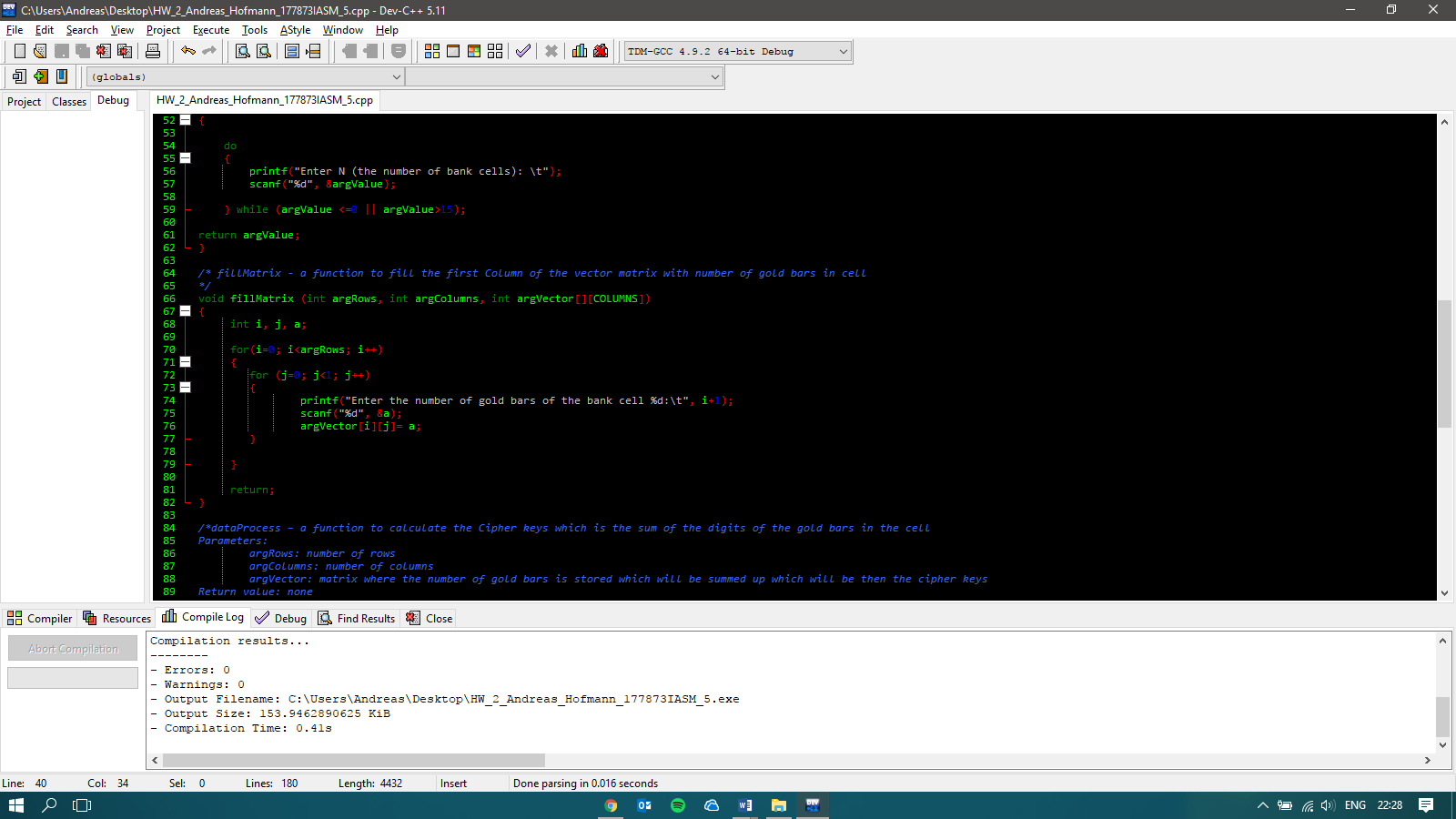
# Snapshots



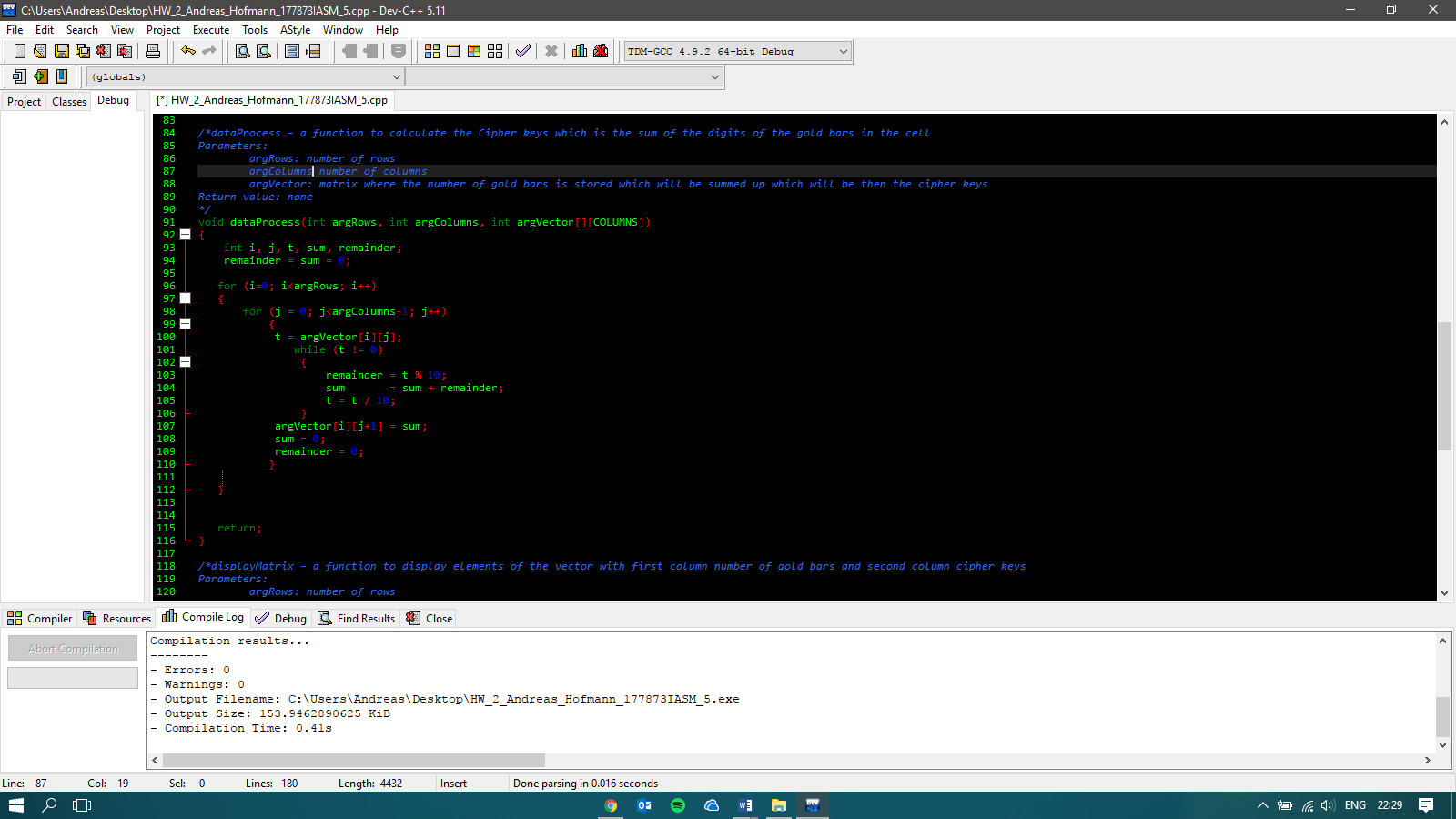
This snapshot shows the header of the program, the libraries included, and the prototypes of the functions will be shown more detailed in later snapshots.



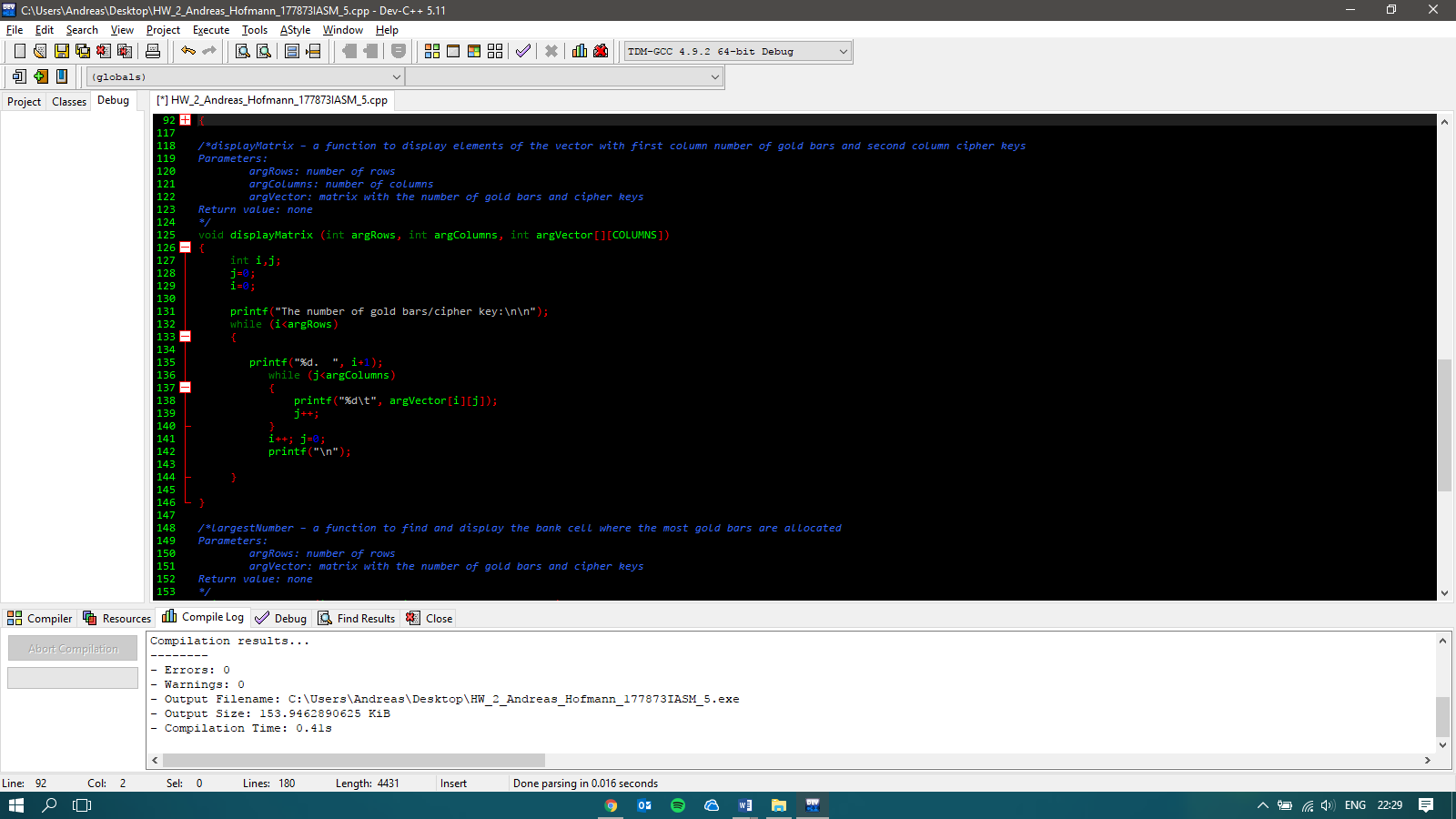
In this snapshot the main function and the function to check the range of the bank cells is shown in detail. Furthermore, the description of the checkRange function can be seen.



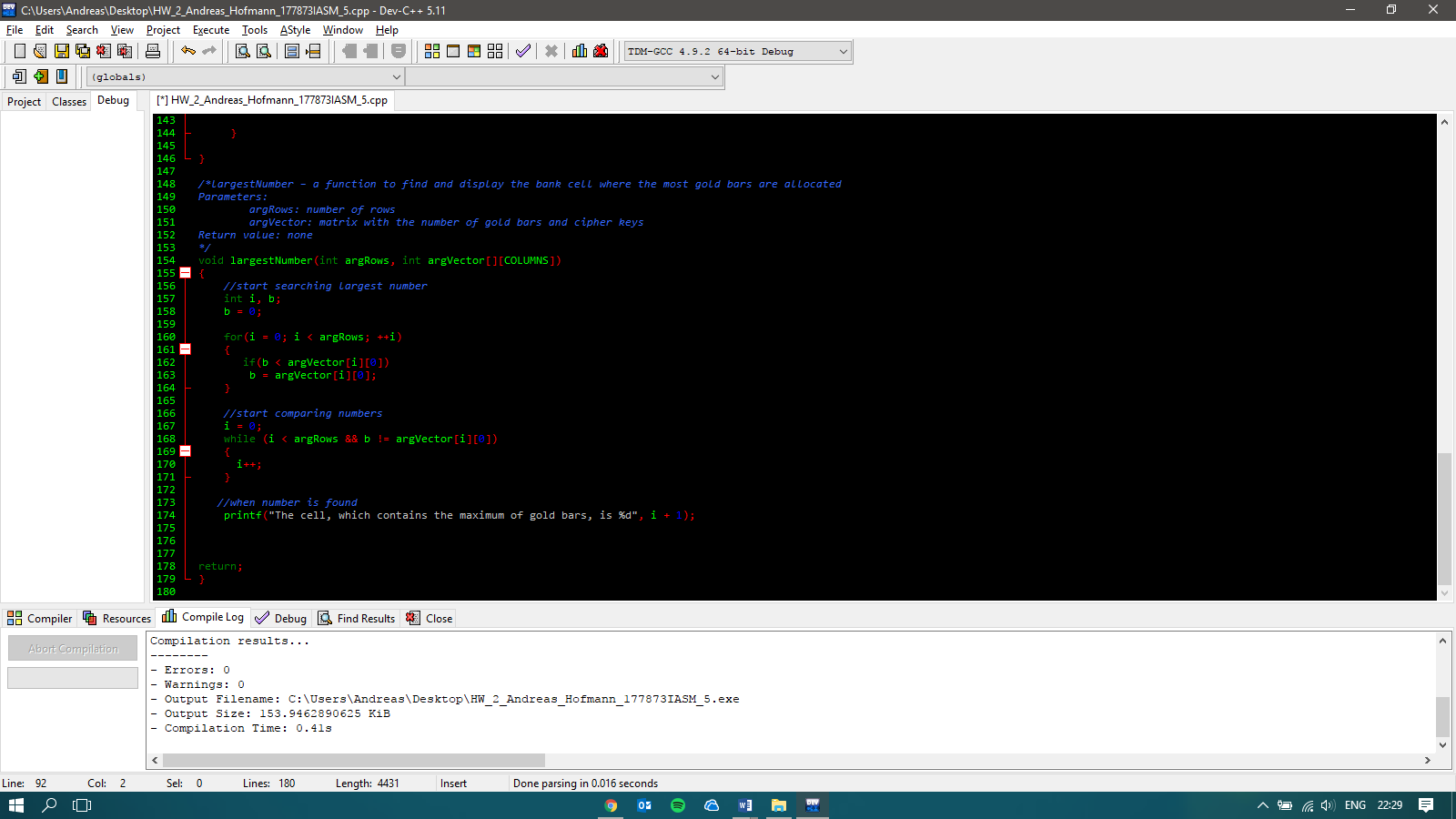
In this snapshot the importance lays on the detailed description of the function ‘fillMatrix’



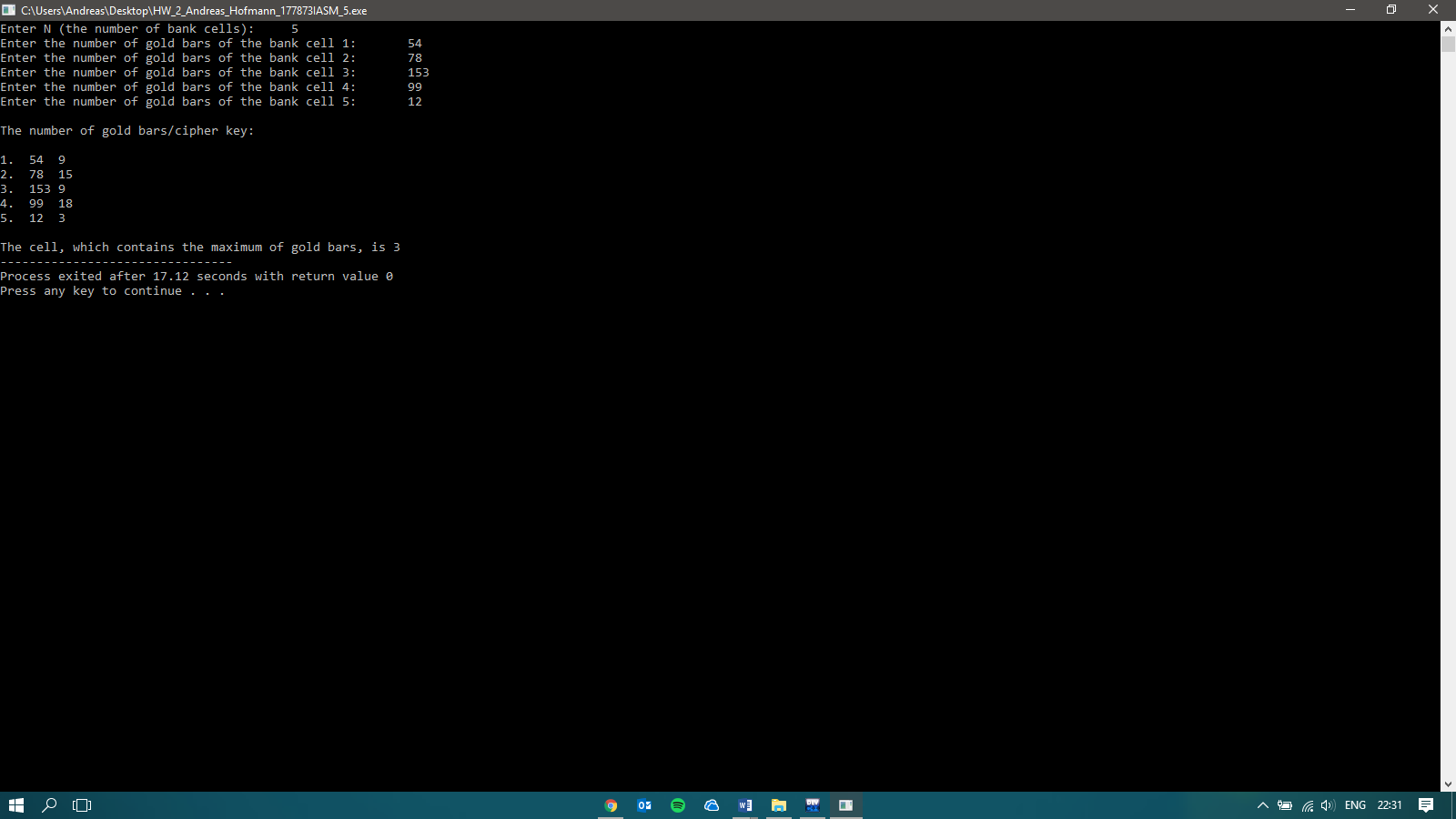
Here the detailed description of the ‘dataProcess’function.



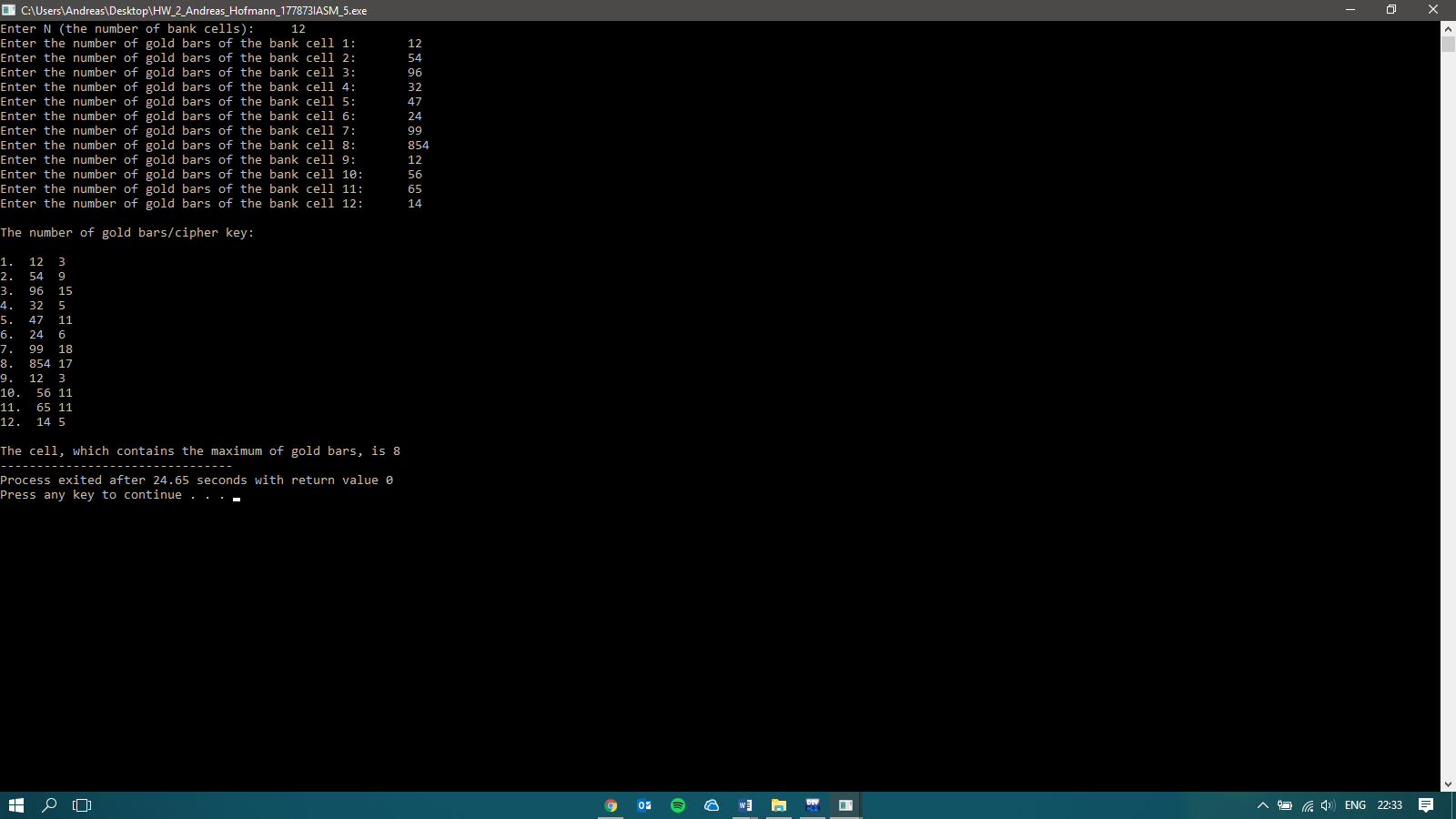
The ‘displayMatrix’ function with its corresponding description



The function to find the largest number of gold bars and its corresponding bank cell.



Execution of the program with 5 bank cells and random numbers.



Execution of the program with 12 bank cells and random numbers.

# Structure of test file with user-based input data

In my two .txt files I used the same numbers as in the two snapshots above, in order to verify that I got the same result.

The first value in the .txt file is the number of bank cells.

The further values should first of all be equal to the amount of bank cells, that means if there are 5 bank cells only 5 numbers should follow. If there are less it is not a big problem since the program will then automatically stop and not use the following values. However, if the amount of bank cells is higher than the following numbers (bank cells is 7 and only 5 values follow), they program will use the last number to fill up the empty spaces.

The following numbers are the amount of gold bars in one cell.

If you have 5 bank cells in total there are 6 numbers (bank cell + gold bars per bank cell).