School of Engineering and Digital Sciences

2048

Sliding Tiles Puzzle

A project submitted

in partial fulfillment of the requirements for the ENG 101 course in SEDS

by

Kosshybay Sanzhar (202016119)

Tuleshov Bektaiyr (202054383)

Kurpetayev Ilyas (202049943)

Damir Abiken (202057936)

Evaluated by

Fall 2022

ACKNOWLEDGEMENT

We would like to acknowledge the tutors of this programming course, Dr. Nursultan Kabylkas, Dr. Aresh Dadlani and Dr. Carlo Molardi for introducing us to the C++ programming language and giving us the necessary base to learn it by ourselves.

UNDERTAKING

This is to declare that the project entitled “2048 Sliding Tiles Puzzle” is an original work done by undersigned, in partial fulfillment of the requirements for the course entitled “ENG 101: Programming for Engineers” at the School of Engineering and Digital Sciences (SEDS), Nazarbayev University.

All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or university.

ABSTRACT:

The final project we have to complete for this assignment is the "2048 Puzzle Game". It required us to use the CodeBlocks IDLE system to implement the code written in the C++ programming language. The purpose of this assignment is to provide the logical conclusion for our study during the semester by programming the project that will include all functions, classes and OOP structures we learned. The work was done exactly according to the criteria set in the project manual. In the course of our work, we relied on the basic materials provided to us in the programming course, including lectures, resources provided by teachers. To make the project more advanced and unique, we also managed to add couple of new functions, previously not studied during the course

Fall 2022

Table of contents

1.0 Introduction:

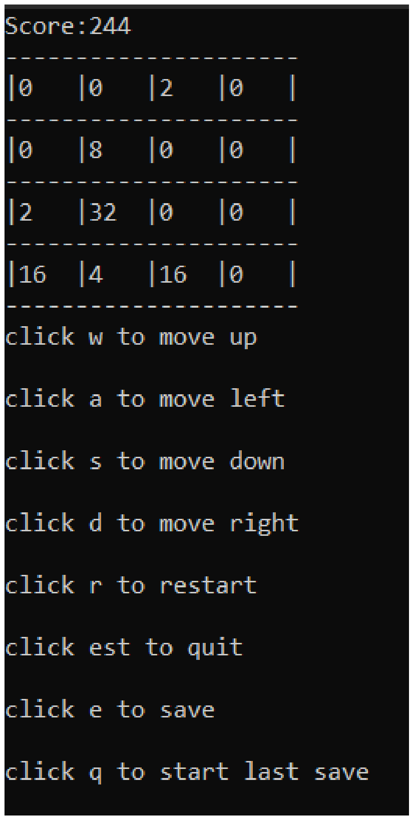
The main goal of this report is to demonstrate the full progress done on the "2048 Sliding Tiles Puzzle" project, providing a step-by-step explanation of all the methods that were involved in creating the game. This project was aimed to test and advance our programming skills obtained during the course, in particular, the array implementation and the work with its elements using the loop functions . However, the concepts in working with object-oriented programming (OOP) were used to implement the following functions: 1) console based interface 2) input interaction from the keyboard 3) score calculator 4) “failure” and “win” detector 5) “restart” function 6) “quit” and “save” function.

2.0 Project Description:

2048 is the puzzle game, based on the addition of numbers that are degrees of 2. There are 16 cells on the two-dimensional plane (4x4) that are empty in the beginning of the game. The game starts with two blocks, commonly the combination of 2 and 4 or the double 2. Then after the movement in one of the four directions the same particles standing at the line of movement add together, while an additional particle is generated at the random place. The system continues to generate new particles, which should be added to the previous, so that the nominal value of them will increase geometrically. The nominal of all added numbers through the game is stored in a specific variable, called score. The game finishes either when all of the cells are full and there is no available movement or if the score reaches a value of 2048.

3.0 Content

3.1 Interface:



```
Score:244
---------------------
|0   |0   |2   |0   |
---------------------
|0   |8   |0   |0   |
---------------------
|2   |32  |0   |0   |
---------------------
|16  |4   |16  |0   |
---------------------
click w to move up

click a to move left

click s to move down

click d to move right

click r to restart

click est to quit

click e to save

click q to start last save
```

The interface presents the score of the previous steps at the upper side. Then the game itself is represented in a box of 16 numbers. In the following part the instructions for the game are provided to the user.

3.1.2 Advanced Features:

The project requirements were to build a 2048 game that will add the numbers, scoring the result of this addition, interact with the user through the input, end game, save and restart functions. In addition to that we wanted to make an advanced interface with border lines between the numbers. The borders here are designed to locate at one place no matter the size of numbers in every cell. This is implemented by using the following code:

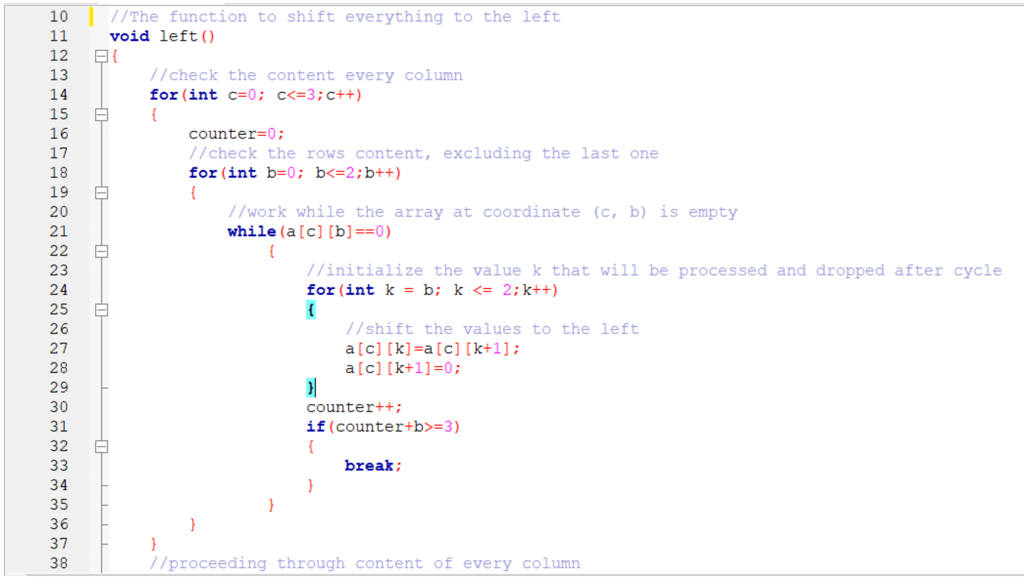
```
//process for each element of an array
for (int i=0; i<4; i++)
{
    //create a border between numbers
    cout << "|";
    for (int j=0; j<4; j++)
    {
        //if the number is small the border will be far away
        if (a[i][j]>=0 && a[i][j]<10)
        {
            //output the number and border together
            cout << a[i][j];
            cout << "   |";
        }
        //if the number is double-digit, i.g. 16, 32, 64 the border will be closer
        else if (a[i][j]>10 && a[i][j]<100)
        {
            //Again output of value and border
            cout << a[i][j];
            cout << "  |";
        }
        //for the large 3-digit number the border will be way closer
        else if (a[i][j]>100 && a[i][j]<1000)
        {
            //output array value and border
            cout << a[i][j];
            cout << " |";
        }
```

3.2 Input interaction from keyboard

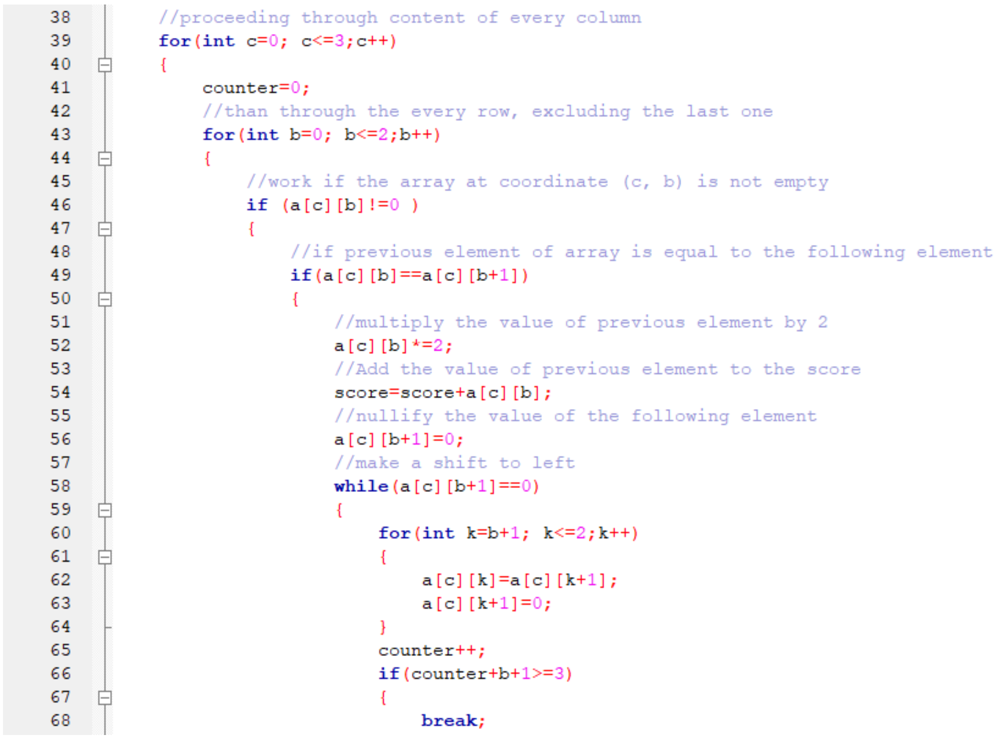
To implement the keyboard interaction we have written the code that moves tiles to the ***"up",*** ***"right"***, ***"left"*** and ***"down"*** sides, according to the key pressed. This code works on the basis of the ***"if"*** and ***"while"*** basic loops functions, that in turn checks the content of each column and row contained in the array. More detailed description can be found in the examples below.

3.2.1 "Left" function

Firstly all non-zero elements are shifted to the left and after that code checks if it is possible to sum the tiles with each other and a new tile with increased value and nonzero tile will appear on the gameboard. If its summed number, it shifts non-zero tiles once again.

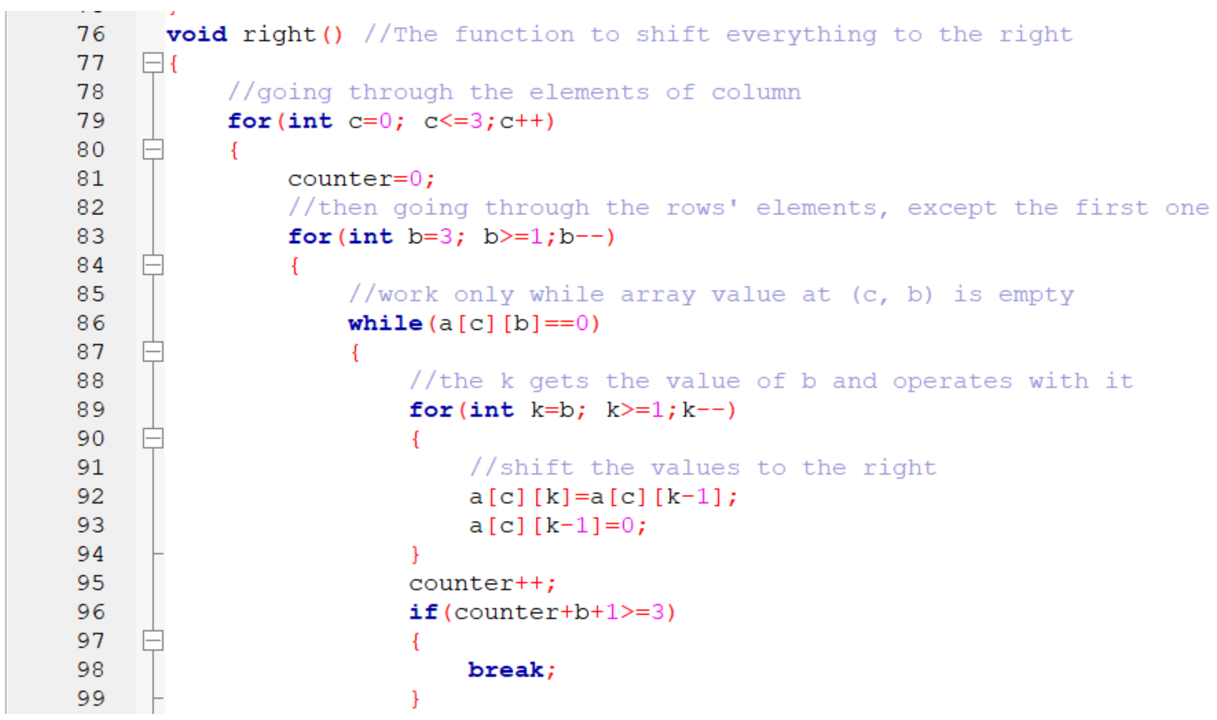


```
//The function to shift everything to the left
void left()
{
    //check the content every column
    for(int c=0; c<=3;c++)
    {
        counter=0;
        //check the rows content, excluding the last one
        for(int b=0; b<=2;b++)
        {
            //work while the array at coordinate (c, b) is empty
            while(a[c][b]==0)
            {
                //initialize the value k that will be processed and dropped after cycle
                for(int k = b; k <= 2;k++)
                {
                    //shift the values to the left
                    a[c][k]=a[c][k+1];
                    a[c][k+1]=0;
                }
                counter++;
                if(counter+b>=3)
                {
                    break;
                }
            }
        }
    }
    //proceeding through content of every column
```



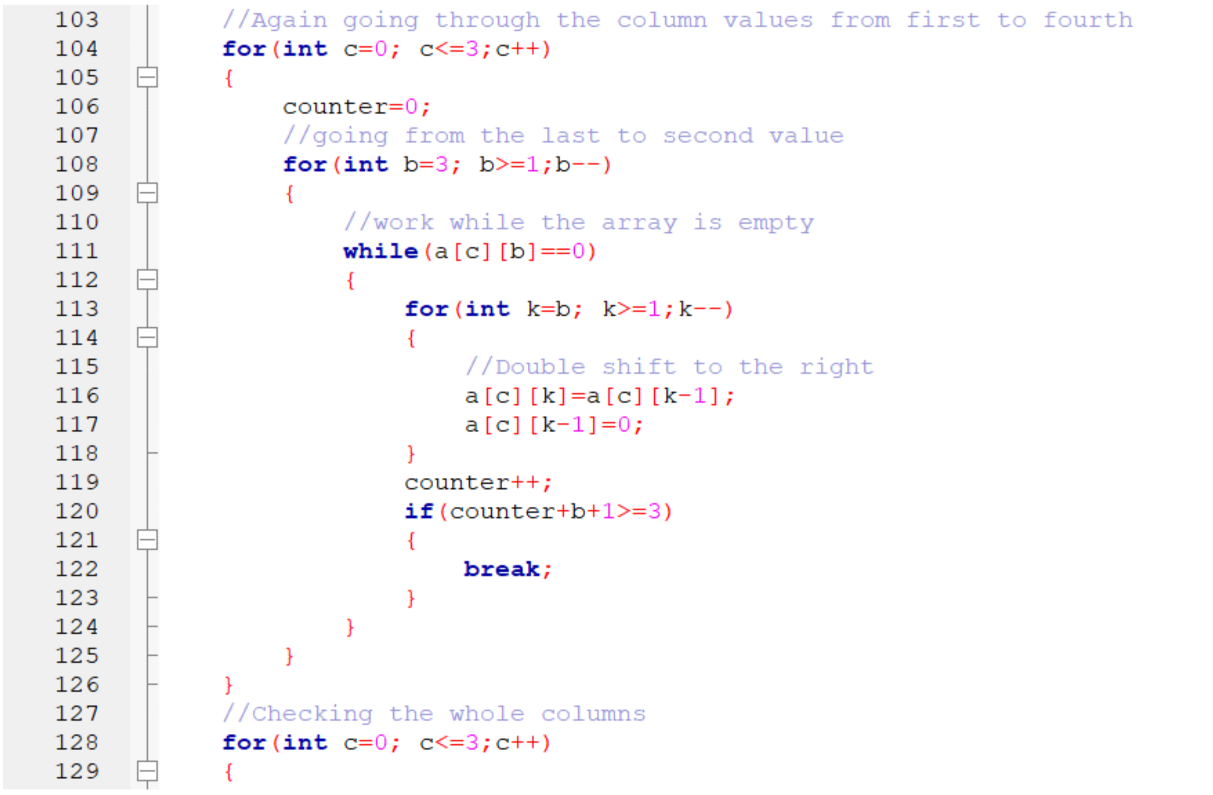
```
    //proceeding through content of every column
    for(int c=0; c<=3;c++)
    {
        counter=0;
        //than through the every row, excluding the last one
        for(int b=0; b<=2;b++)
        {
            //work if the array at coordinate (c, b) is not empty
            if (a[c][b]!=0 )
            {
                //if previous element of array is equal to the following element
                if(a[c][b]==a[c][b+1])
                {
                    //multiply the value of previous element by 2
                    a[c][b]*=2;
                    //Add the value of previous element to the score
                    score=score+a[c][b];
                    //nullify the value of the following element
                    a[c][b+1]=0;
                    //make a shift to left
                    while(a[c][b+1]==0)
                    {
                        for(int k=b+1; k<=2;k++)
                        {
                            a[c][k]=a[c][k+1];
                            a[c][k+1]=0;
                        }
                        counter++;
                        if(counter+b+1>=3)
                        {
                            break;
```

3.2.2. “Right” function

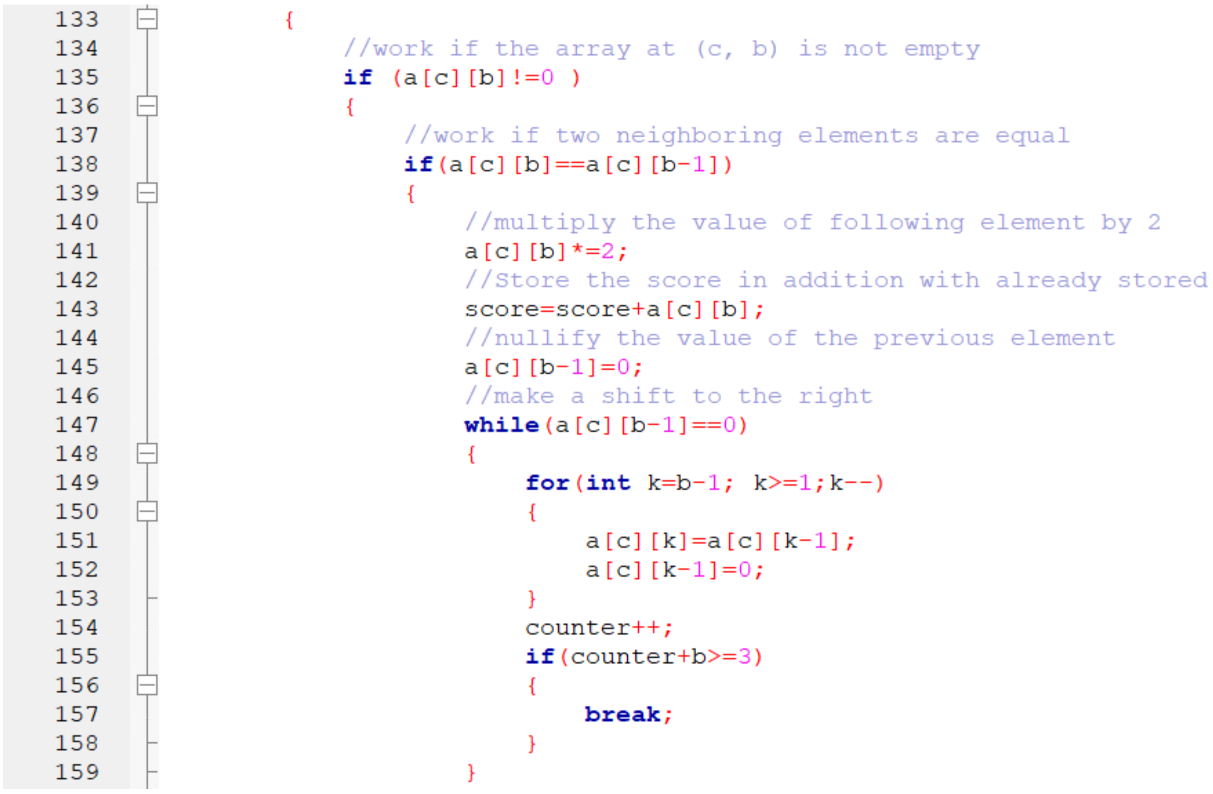


```
void right() //The function to shift everything to the right
{
    //going through the elements of column
    for(int c=0; c<=3;c++)
    {
        counter=0;
        //then going through the rows' elements, except the first one
        for(int b=3; b>=1;b--)
        {
            //work only while array value at (c, b) is empty
            while(a[c][b]==0)
            {
                //the k gets the value of b and operates with it
                for(int k=b; k>=1;k--)
                {
                    //shift the values to the right
                    a[c][k]=a[c][k-1];
                    a[c][k-1]=0;
                }
                counter++;
                if(counter+b+1>=3)
                {
                    break;
                }
```

The “right” function mostly works with the same principle as the “Left” function except that the checking of elements of an array goes from right to left. In addition, the movement of elements is directed to the right. Also, it shifts twice in comparison with left were non zero elements were shifted only once.

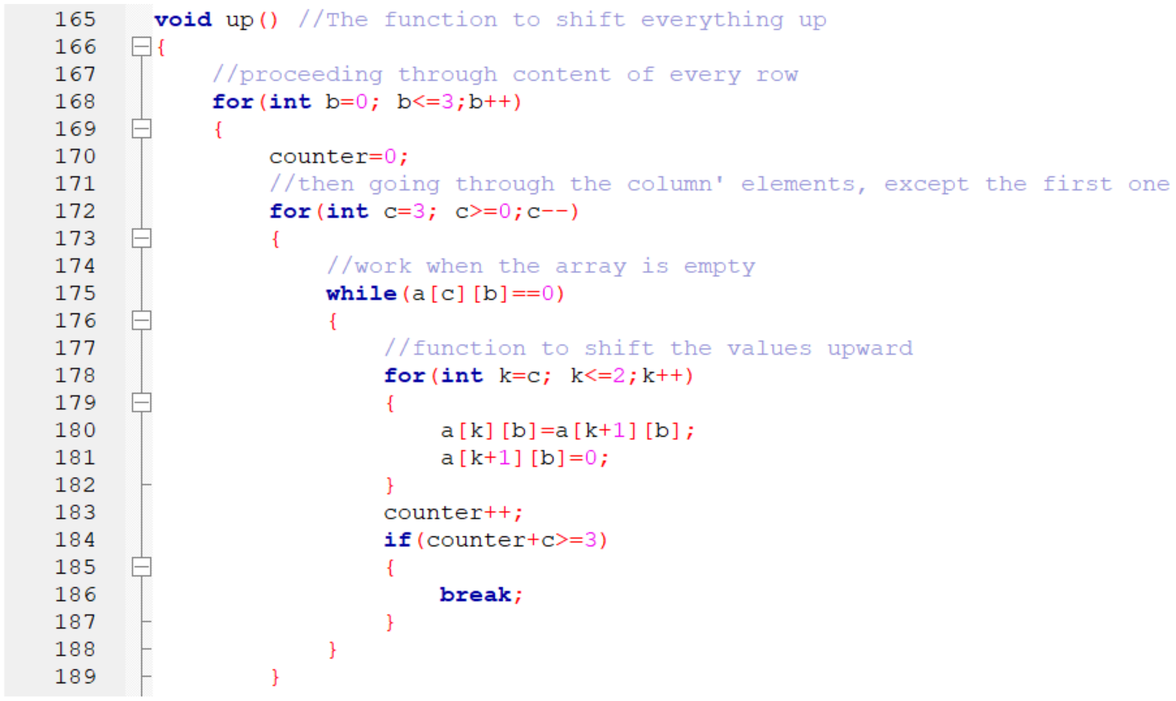


```
    //Again going through the column values from first to fourth
    for(int c=0; c<=3;c++)
    {
        counter=0;
        //going from the last to second value
        for(int b=3; b>=1;b--)
        {
            //work while the array is empty
            while(a[c][b]==0)
            {
                for(int k=b; k>=1;k--)
                {
                    //Double shift to the right
                    a[c][k]=a[c][k-1];
                    a[c][k-1]=0;
                }
                counter++;
                if(counter+b+1>=3)
                {
                    break;
                }
            }
        }
    }
    //Checking the whole columns
    for(int c=0; c<=3;c++)
    {
```

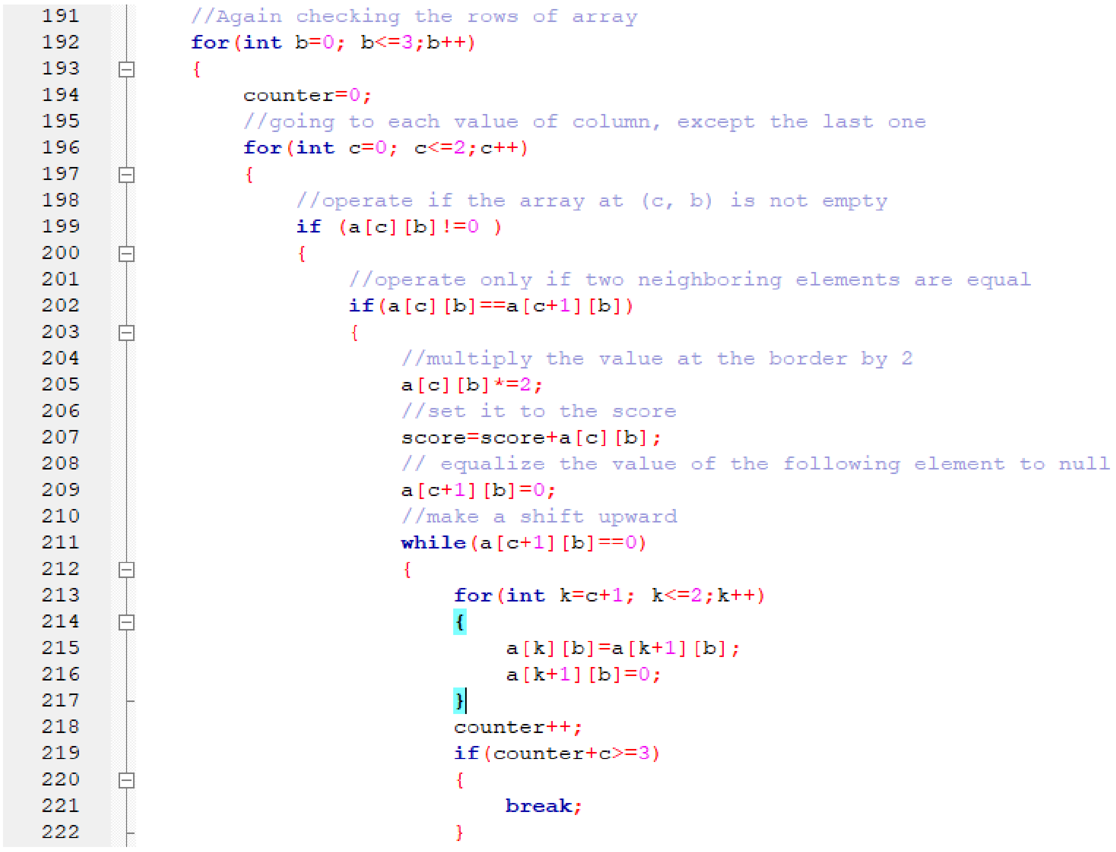


```
        {
            //work if the array at (c, b) is not empty
            if (a[c][b]!=0 )
            {
                //work if two neighboring elements are equal
                if(a[c][b]==a[c][b-1])
                {
                    //multiply the value of following element by 2
                    a[c][b]*=2;
                    //Store the score in addition with already stored
                    score=score+a[c][b];
                    //nullify the value of the previous element
                    a[c][b-1]=0;
                    //make a shift to the right
                    while(a[c][b-1]==0)
                    {
                        for(int k=b-1; k>=1;k--)
                        {
                            a[c][k]=a[c][k-1];
                            a[c][k-1]=0;
                        }
                        counter++;
                        if(counter+b>=3)
                        {
                            break;
                        }
                    }
```

3.2.3 “Up” function:

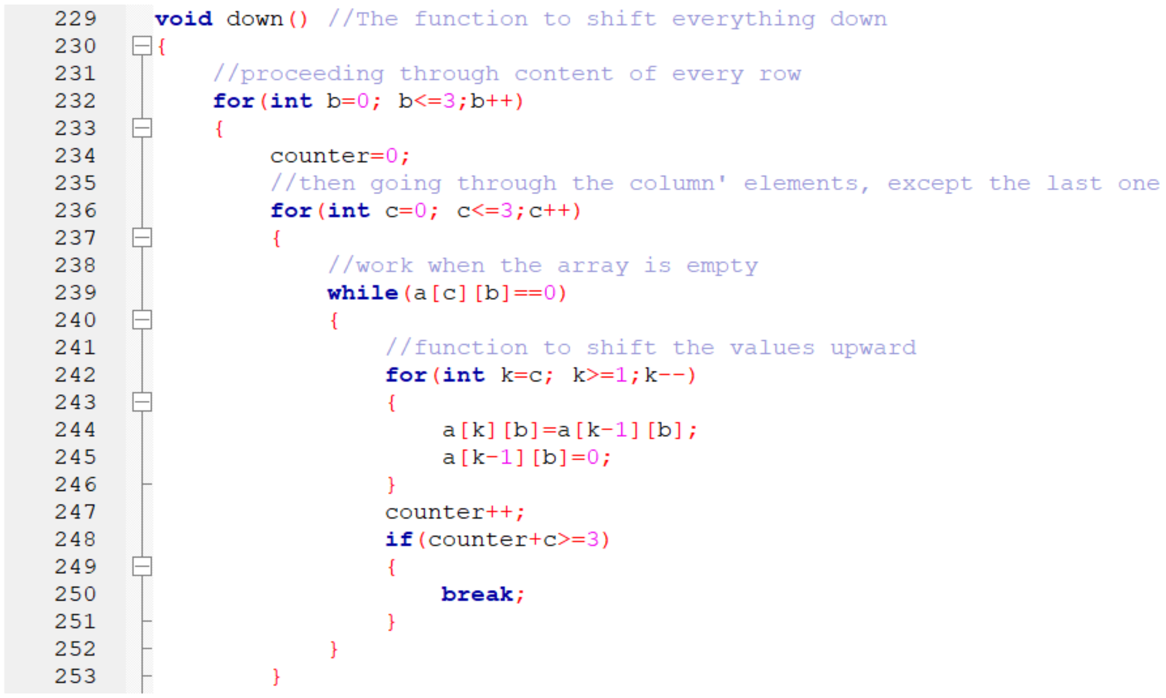


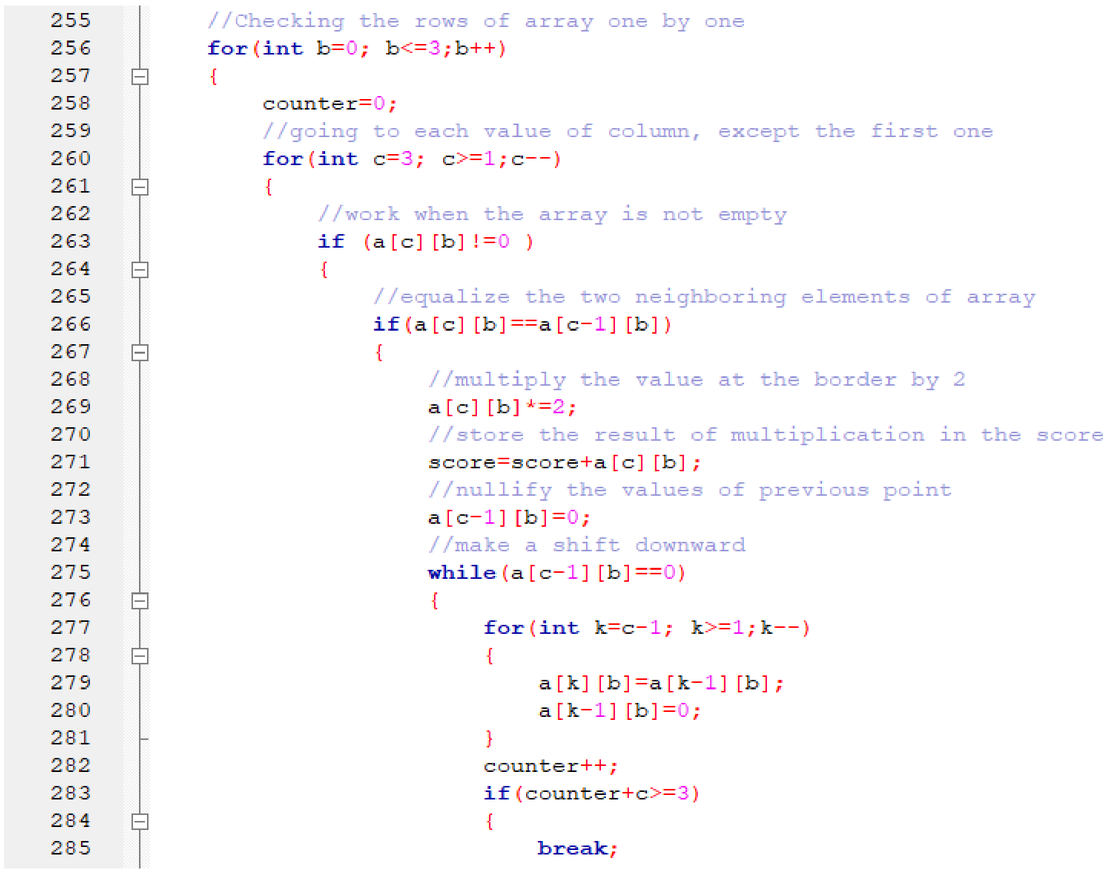
```
void up() //The function to shift everything up
{
    //proceeding through content of every row
    for(int b=0; b<=3;b++)
    {
        counter=0;
        //then going through the column' elements, except the first one
        for(int c=3; c>=0;c--)
        {
            //work when the array is empty
            while(a[c][b]==0)
            {
                //function to shift the values upward
                for(int k=c; k<=2;k++)
                {
                    a[k][b]=a[k+1][b];
                    a[k+1][b]=0;
                }
                counter++;
                if(counter+c>=3)
                {
                    break;
                }
            }
        }
```



```
    //Again checking the rows of array
    for(int b=0; b<=3;b++)
    {
        counter=0;
        //going to each value of column, except the last one
        for(int c=0; c<=2;c++)
        {
            //operate if the array at (c, b) is not empty
            if (a[c][b]!=0 )
            {
                //operate only if two neighboring elements are equal
                if(a[c][b]==a[c+1][b])
                {
                    //multiply the value at the border by 2
                    a[c][b]*=2;
                    //set it to the score
                    score=score+a[c][b];
                    // equalize the value of the following element to null
                    a[c+1][b]=0;
                    //make a shift upward
                    while(a[c+1][b]==0)
                    {
                        for(int k=c+1; k<=2;k++)
                        {
                            a[k][b]=a[k+1][b];
                            a[k+1][b]=0;
                        }
                        counter++;
                        if(counter+c>=3)
                        {
                            break;
                        }
```

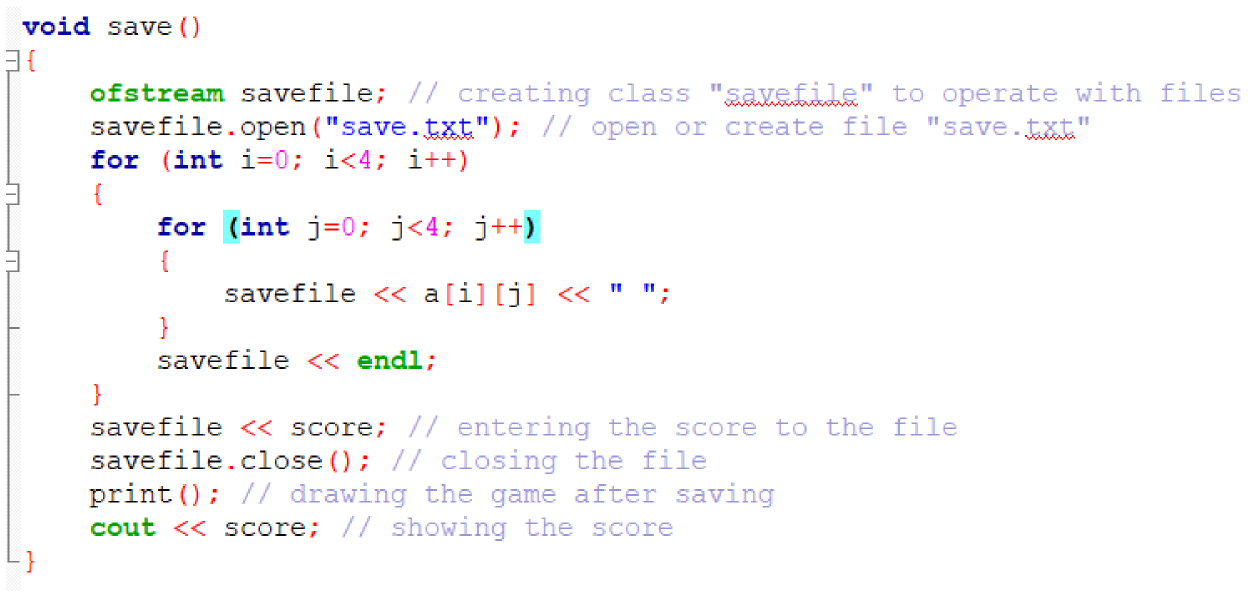
3.2.4 “Down” function


```
void down() //The function to shift everything down
{
    //proceeding through content of every row
    for(int b=0; b<=3;b++)
    {
        counter=0;
        //then going through the column' elements, except the last one
        for(int c=0; c<=3;c++)
        {
            //work when the array is empty
            while(a[c][b]==0)
            {
                //function to shift the values upward
                for(int k=c; k>=1;k--)
                {
                    a[k][b]=a[k-1][b];
                    a[k-1][b]=0;
                }
                counter++;
                if(counter+c>=3)
                {
                    break;
                }
            }
        }
```


```
    //Checking the rows of array one by one
    for(int b=0; b<=3;b++)
    {
        counter=0;
        //going to each value of column, except the first one
        for(int c=3; c>=1;c--)
        {
            //work when the array is not empty
            if (a[c][b]!=0 )
            {
                //equalize the two neighboring elements of array
                if(a[c][b]==a[c-1][b])
                {
                    //multiply the value at the border by 2
                    a[c][b]*=2;
                    //store the result of multiplication in the score
                    score=score+a[c][b];
                    //nullify the values of previous point
                    a[c-1][b]=0;
                    //make a shift downward
                    while(a[c-1][b]==0)
                    {
                        for(int k=c-1; k>=1;k--)
                        {
                            a[k][b]=a[k-1][b];
                            a[k-1][b]=0;
                        }
                        counter++;
                        if(counter+c>=3)
                        {
                            break;
```

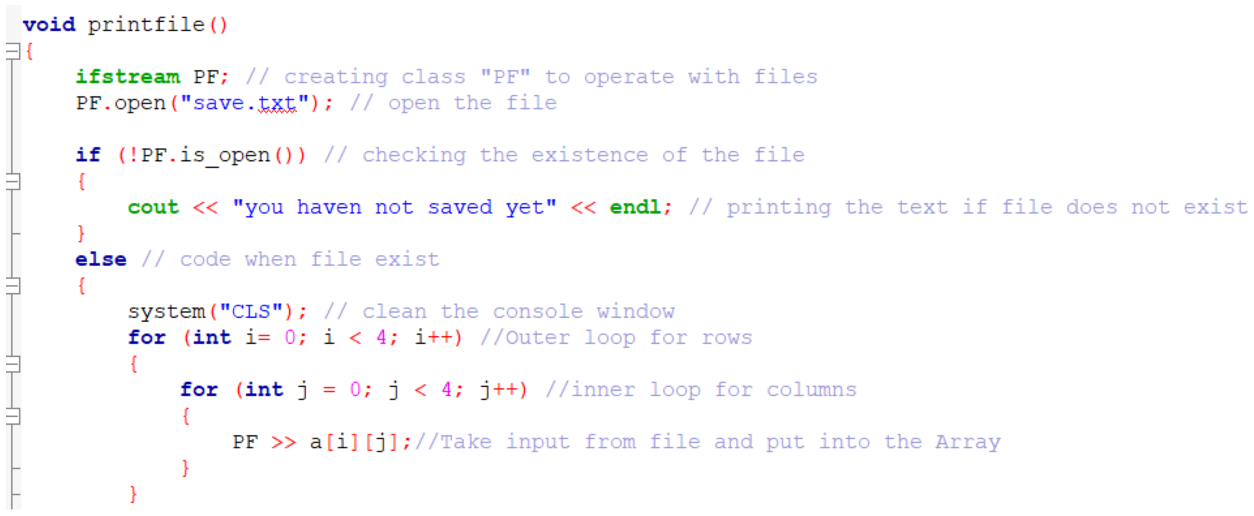
3.2.5 “Save the game” and “Load last saved game” functions:

For “Save the game” function we create a new class called “savefile” that will operate with the outer files.



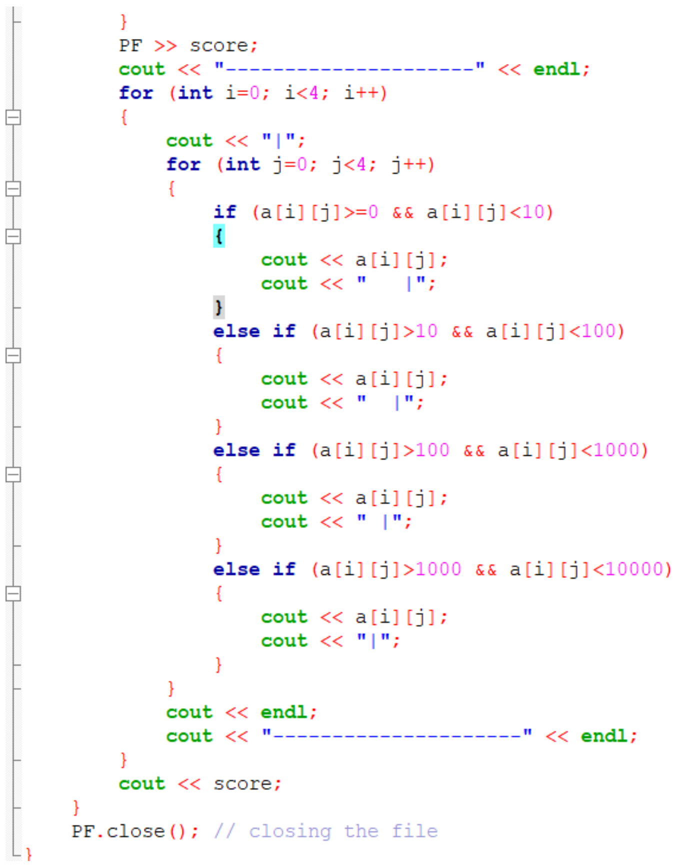
```
void save()
{
    ofstream savefile; // creating class "savefile" to operate with files
    savefile.open("save.txt"); // open or create file "save.txt"
    for (int i=0; i<4; i++)
    {
        for (int j=0; j<4; j++)
        {
            savefile << a[i][j] << " ";
        }
        savefile << endl;
    }
    savefile << score; // entering the score to the file
    savefile.close(); // closing the file
    print(); // drawing the game after saving
    cout << score; // showing the score
}
```

The printfile function takes the data from the save function and penetrates it into an array.



```
void printfile()
{
    ifstream PF; // creating class "PF" to operate with files
    PF.open("save.txt"); // open the file

    if (!PF.is_open()) // checking the existence of the file
    {
        cout << "you haven not saved yet" << endl; // printing the text if file does not exist
    }
    else // code when file exist
    {
        system("CLS"); // clean the console window
        for (int i= 0; i < 4; i++) //Outer loop for rows
        {
            for (int j = 0; j < 4; j++) //inner loop for columns
            {
                PF >> a[i][j];//Take input from file and put into the Array
            }
        }
```

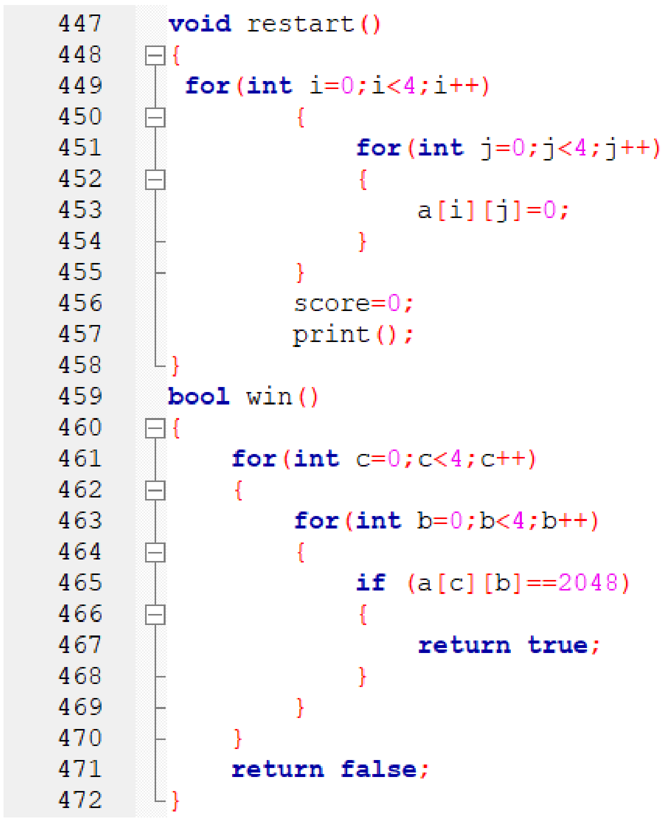


```
        }
        PF >> score;
        cout << "---------------------" << endl;
        for (int i=0; i<4; i++)
        {
            cout << "|";
            for (int j=0; j<4; j++)
            {
                if (a[i][j]>=0 && a[i][j]<10)
                {
                    cout << a[i][j];
                    cout << "   |";
                }
                else if (a[i][j]>10 && a[i][j]<100)
                {
                    cout << a[i][j];
                    cout << "  |";
                }
                else if (a[i][j]>100 && a[i][j]<1000)
                {
                    cout << a[i][j];
                    cout << " |";
                }
                else if (a[i][j]>1000 && a[i][j]<10000)
                {
                    cout << a[i][j];
                    cout << "|";
                }
            }
            cout << endl;
            cout << "---------------------" << endl;
        }
        cout << score;
    }
    PF.close(); // closing the file
}
```

3.2.6 "Quit the game" function :

```
bool gameOver(int row, int column)
{
    //check all elements of rows
    for(row=0; row<=3;row++)
    {
        //check starting from first to before last
        for(column=0;column<=2;column++)
        {
            // if the content of two neighboring columns are equal
            if(a[row][column]==a[row][column+1])
            {
                //the function will not be executed
                return false;
            }
        }
    }
    //checking each column
    for(column=0;column<=3;column++)
    {
        //then each row, except the last one
         for(row=0; row<=2;row++)
        {
            //if the content of two neighboring rows is the same
            if(a[row][column]==a[row+1][column])
            {
                //the function will not be executed
                return false;
            }
        }
    }
    //otherwise proceed the function
    return true;
}
```

3.2.7 “Restart the game” function:



```
void restart()
{
 for(int i=0;i<4;i++)
        {
            for(int j=0;j<4;j++)
            {
                a[i][j]=0;
            }
        }
        score=0;
        print();
}
bool win()
{
    for(int c=0;c<4;c++)
    {
        for(int b=0;b<4;b++)
        {
            if (a[c][b]==2048)
            {
                return true;
            }
        }
    }
    return false;
}
```

4.0 Main Challenges:

During the coding process, we have faced a lot of difficulties that changed our approach towards writing and pushed the implementation of new ideas. There were issues with the “right” function and function “nn”.

The “Right” function had a bug, where the shift was not always complete. For that reason, the piece of code that shifts tiles was done twice in order to fix this problem. However, in functions “Up”, “Down” and “Left” it was implemented only once.

The new numbers were appearing even if we used non-shifting functions. To deal with that problem we created a “nn” function that restricts the appearance of unnecessary numbers.

Another problem occurred with a “Save and Load” function that has been incorrectly loading data from the file, until we fixed it.

5.0 Conclusion:

To conclude our team “Hello World” met the arranged goals and successfully designed the 2048 Puzzle Game. Code for the project was built in accordance with the project manual, while there were additions made with interface designed by our team. Although we faced problems during the programming stage of a project, they were consequently solved by team effort. The project helped us to deeply understand the C++ programming language and improve our skills in coding.

6.0 Major Contribution:

The programming and designing the game was made by joint effort of every member of a team. There were some challenges we encountered that broke the code and outputted a lot of bags. We overcame them using collaborative work to generate new sufficient ideas. In addition every team member tried to implement an equal contribution to the code and report. Although the main contributions to code were made by Sanzhar and Bektaiyr, the minor functions were implemented by Damir and Ilyas. The OOP was designed by Bektaiyr, in order to make an A-level code. The changing in accordance with the coding guidelines was made by Ilyas Kurpetayev