**Assignment5**

**exercise 1**

Question

Write a C function which has the prototype int is\_loweralpha(char \* str);

The function should check if a string pointed to by str contains onlythe lower case letter

characters (a-z).

The return value of the function should be:

1 if all of the characters are lower case letters (a-z)

0 if any of the characters are not the lower case letters (a-z)

Model Answer

Software Development Process

1. **Problem statement**

To ask user to input a sentence and then check this sentence, if all words in the sentence are lowercase letters, display 1, if any words are not lowercase include symbols and digits, display 0.

1. **Analysis**

Inputs:

User enters a string in program.

Outputs:

If all characters are lower case letters, display 1, else, display 0.

Additional requirements or constraint

No.

**3. Design**

Algorithm

1. Adding ‘stdio.h’ and ‘ctype.h’ library.
2. (1)Create defined function “is\_loweralpha”

(2)Char \*str-represents entered characters.

(3)int i,j-for counting.

(4)Ask user to enter characters which need to check.

(5) Read and store entered characters using gets function.

(6)Check entered strings using islower function in ctype library.

(7)Count strings, and then count number of lower letters.

(8)Compare i and j, if they are same, return 1, if they are not same, return 0.

1. char a[666666]-represents entered strings

int b-represents return value.

4. Display and tell user how to run this program.

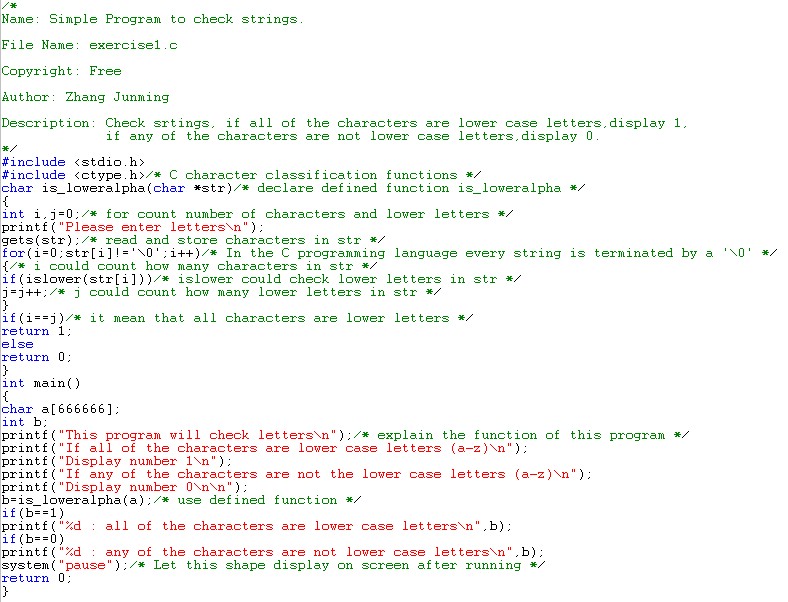
5.  Use defined function “is\_loweralpha” check characters in a[666666].

6. Compare the value of b.

7. Display the result on the screen.

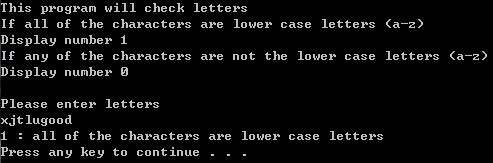
1. **Implementation**:

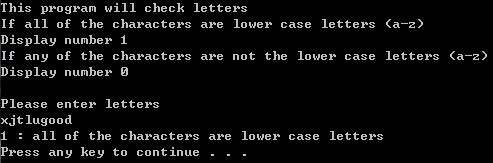
See the C code in file exercise1.c with comments.



1. **Testing:**

The C program was tested by carrying out a set of experiments; and the C program output was verified successfully. For instance,





**Exercises2**

Question

Write a C program which can perform the following operations on a 4x4 matrix, where each operation (1-8) is performed by a separate function; menu option 9 does not require a function:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Matrix Operation\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*(1)initialize the matrix with a value

\*(2)fill each matrix element with entered values

\*(3)rotate the matrix clockwise or anti-clockwise

\*(4)transpose the matrix

\*(5)fill the matrix with random values from -20 to 30

\*(6)display the current matrix

\*(7)fill the outer elements of the matrix with a value

\*(8)fill the inner elements of the matrix with a value

\*(9)quit

Model Answer

Software Development Process

1. **Problem statement**

Ask user to choose functions using number (1~9), 1~9 represent different function of matrix: 1- initialize the matrix with a value; 2- fill each matrix element with entered values; 3- rotate the matrix clockwise or anti-clockwise; 4- transpose the matrix; 5- fill the matrix with random values from -20 to 30; 6- display the current matrix; 7- fill the outer elements of the matrix with a value;

8- fill the inner elements of the matrix with a value; 9-quit.

1. **Analysis**

Inputs:

Numbers (1~9) choose function and the value of matrix.

Outputs:

Different required matrix.

Additional requirements or constraints:

Prototype of defined function is: void \*\*\*\*\_mx (int (\*mx)[SIZE\_COLUMN])

**3. Design**

Algorithm

1. Adding ‘stdio.h’, ’stdlib.h’ and ‘time.h’ library.
2. Define SIZE\_COLUMN and SIZE\_ROW are 4.
3. Create first defined function ‘menu ()’:

(1)Ask user to input number to choose functions.

(2)Display the main menu.

4. Create second defined function, ‘init\_mx(int (\*matrix)[SIZE\_COLUMN])’

(1)int i,j-represent row and column.

(2)int a-represents initialized value.

(3)Ask user to input initialized value.

(4)Read and store this value.

(5)Setting up a loop using ‘for’ function, let 16 elements equal to initialized value.

(6)Setting up a loop using ‘for’ function, display this matrix.

(7)Use system ‘pause’ and ‘cls’ clear screen after using this defined function.

5. Create third defined function, ‘fill\_mx(int (\*matrix)[SIZE\_COLUMN])’

(1) int i,j-represent row and column.

(2)Ask user to input 16 digits to fill the matrix.

(3)Setting up a loop using ‘for’ function, store all entered value in 16 elements.

(4)Setting up a loop using ‘for’ function, display this matrix.

(5)Use system ‘pause’ and ‘cls’ clear screen after using this defined function.

6. Create forth defined function, ‘rot\_mx(int (\*matrix)[SIZE\_COLUMN])’

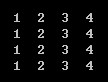
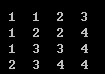
(1)int i,j-represent row and column.

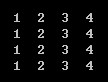
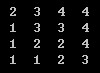
(2)int a-represents two branches.

(3) Setting up a loop using ‘for’ function, display origin matrix.

(4)Ask user choose branch.1 for clockwise and 2 for anti-clockwise.

(5)Read and store value in a.

(6)clockwise: my understanding is same as  to 

anti-clockwise: my understanding is same as  to 

(7)display matrix using printf(a[0][0] to a[0][1]) and so on when choose clockwise, printf(a[0][0] to a[1][0]) and so on when choose anti-clockwise.

(8)Use system ‘pause’ and ‘cls’ clear screen after using this defined function.

7. Create forth defined function, trans\_mx(int (\*matrix)[SIZE\_COLUMN])’

(1)int i,j-represent row and column.

(2) Setting up a loop using ‘for’ function, display origin matrix.

(3) Setting up a loop using ‘for’ function, display transposed matrix when change i to j and j to i.

(4) Use system ‘pause’ and ‘cls’ clear screen after using this defined function.

8. Create third defined function, ‘rand\_mx(int (\*matrix)[SIZE\_COLUMN])’

(1) int i,j-represent row and column.

(2)Reset time using srand((unsigned)time(NULL)) in time.h.

(3)Setting up a loop using ‘for’ function, put random value in each elements, rand function in stdlib.h library. Closed interval is -20 to 30, therefore, 30-(-20)=50, rand()%(50)+(-20) could create random number in -20 to 30.

(4)Setting up a loop using ‘for’ function, display this matrix.

(5)Use system ‘pause’ and ‘cls’ clear screen after using this defined function.

9. Create forth defined function, dis\_mx(int (\*matrix)[SIZE\_COLUMN])’

(1)int i,j-represent row and column.

(2) Setting up a loop using ‘for’ function, display current matrix.

(3) Use system ‘pause’ and ‘cls’ clear screen after using this defined function.

10. Create forth defined function, outer\_mx(int (\*matrix)[SIZE\_COLUMN])’

(1)int i,j-represent row and column.

(2)int a-represents the value of outer elements.

(3)Reset time using srand((unsigned)time(NULL)) in time.h.

(4)Ask user to input one number to fill outer elements.

(5)Read and store this number in a.

(6)Setting up a loop using ‘for’ function and ‘if’ function, fill inner elements using random number, fill outer elements using stored number in a.

(7)Setting up a loop using ‘for’ function, display this matrix.

(8)Use system ‘pause’ and ‘cls’ clear screen after using this defined function.

11. Create forth defined function, inner\_mx(int (\*matrix)[SIZE\_COLUMN])’

(1)int i,j-represent row and column.

(2)int a-represents the value of inner elements.

(3)Reset time using srand((unsigned)time(NULL)) in time.h.

(4)Ask user to input one number to fill inner elements.

(5)Read and store this number in a.

(6)Setting up a loop using ‘for’ function and ‘if’ function, fill outer elements using random number, fill inner elements using stored number in a.

(7)Setting up a loop using ‘for’ function, display this matrix.

(8)Use system ‘pause’ and ‘cls’ clear screen after using this defined function.

10. Setting up main program.

11.int z-represents options.

int matrix1[SIZE\_ROW][SIZE\_COLUMN]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,}

represents initial matrix.

12.Setting up a loop using ‘do while’, ask user to input a number choose functions, only input ‘9’ could out this loop.

(1)Using defined function menu().

(2)Read and store entered value in z.

(3)Setting up ‘switch’ function of z.

Case 1: using init\_mx function of matrix1

Case 2: using fill\_mx function of matrix1

Case 3: using rot\_mx function of matrix1

Case 4: using trans\_mx function of matrix1

Case 5: using rand\_mx function of matrix1

Case 6: using disp\_mx function of matrix1

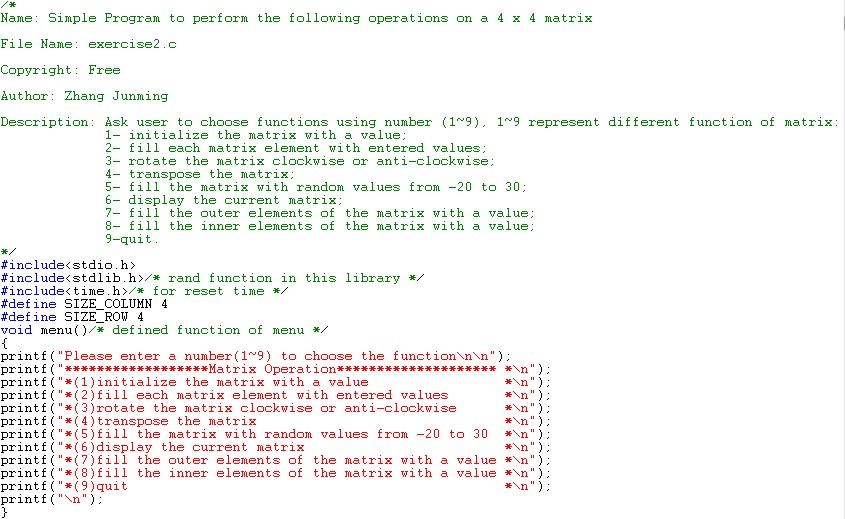
Case 7: using outer\_mx function of matrix1

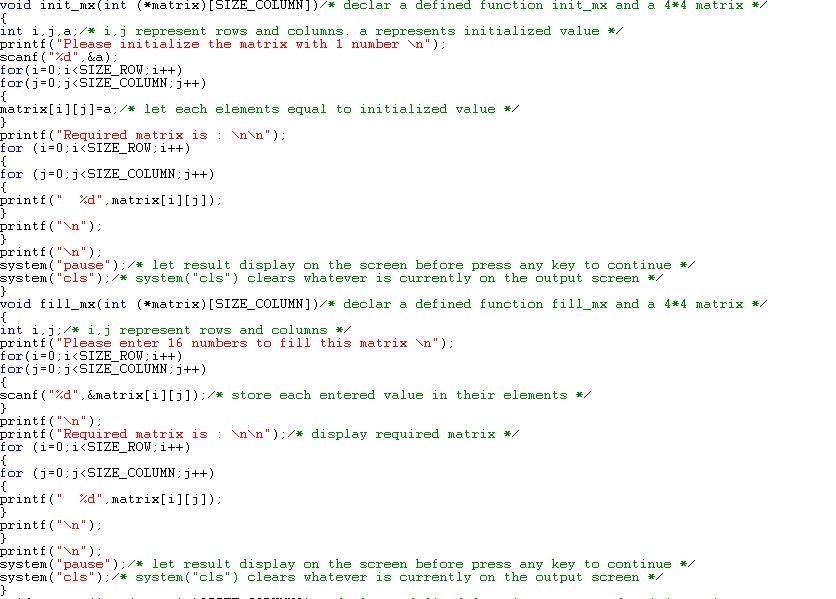
Case 8: using inner\_mx function of matrix1

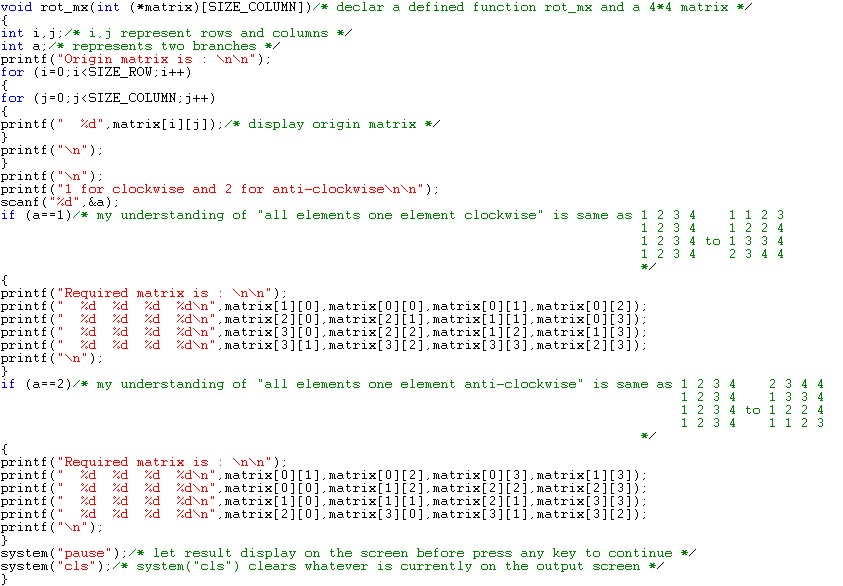
Case 9: noting.

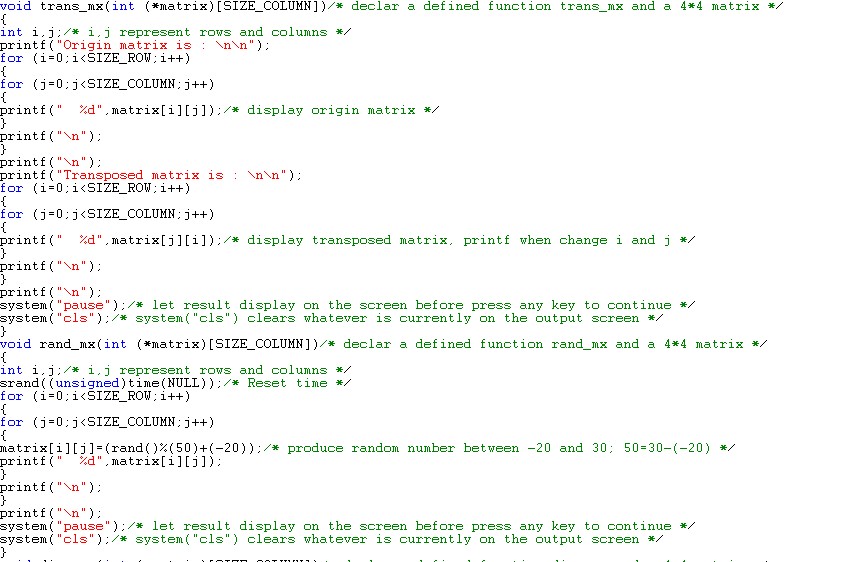
Default when entered number is not in 1~9, display enter data error.

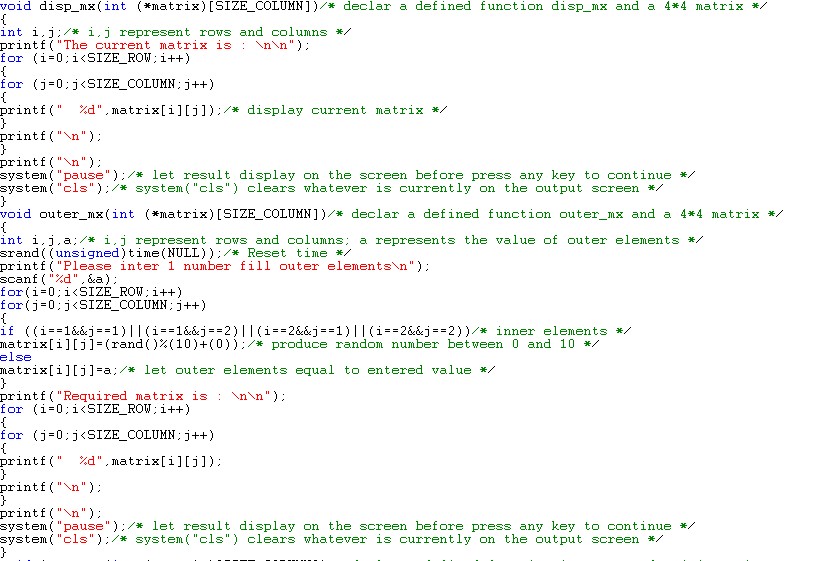
4. Implementation: see the C code in file exercise2.c with comments.

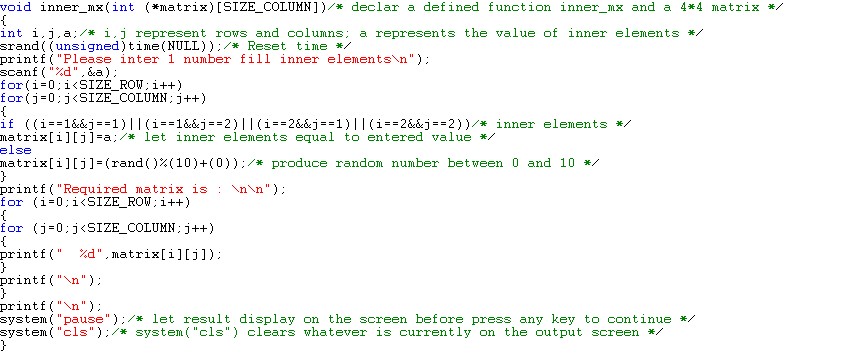


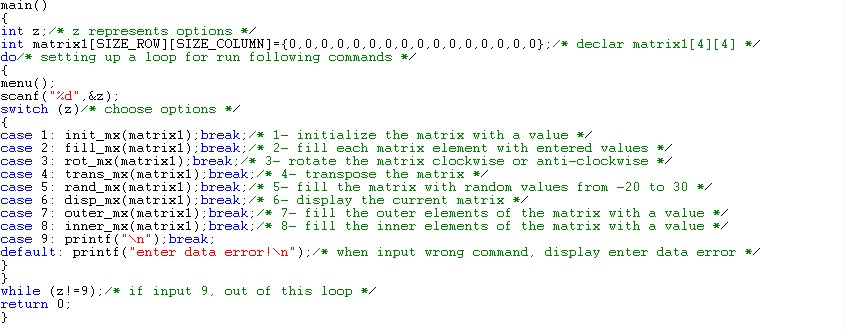












5. Testing:

The C program was tested by carrying out a set of experiments; and the C program output was verified successfully. For instance,

