ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

Implementing all Possible Matrix Operations as Library Function to Enhancement of C Compiler

Pritam Ghosh[1], Sudip Sinha[1], Prof. (Dr.) Pranam Paul[2]

MCA Final Year Student, Narula Institute of Technology, Agarpara, Kolkata, West Bengal, India[1]

HOD, Department of Computer Application, Narula Institute of Technology, Agarpara, Kolkata, West Bengal, India[2]

ABSTRACT: In this paper, we try to add some user define functions in C compiler like printf() , scanf(). Different operations of matrix, which are not present in C – Compiler, are being tried add as library functions. Then to calculate of different kinds of matrix operations can directly be used as calling of functions which is written in byte code. Header file of these functions has also been generated though writing the prototype. Our goal is adding more features in c compiler to make more useful for any programmer, but with maintaining proper concept and process of compiling the code through C compiler.

KEYWORDS: Matrix Addition, Matrix Subtraction, Matrix Multiplication, Determinant, C- Compiler, Library File, Header File, Byte Code, Object File. Linker, Loader,

I. INTRODUCTION

In mathematics, a **matrix** is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns. The mathematical definition of a matrix finds applications in computing and database management, a basic starting point being the concept of arrays. A two-dimensional array can function exactly like a matrix. To illustrate our work let us considered Matrix multiplication, addition, subtraction, determination etc.
In matrix multiplication we use pointer to an array to hold the address of the array value and use undefined array. Here we give any size of array at run time.Matrix representation is a method used by a computer language to store matrices of more than one dimension in memory.When we do some matrix operation then we have to code with mathematical operation; Like as matrix_add, matrix_sub , matrix_mult , matrix_det . Here we created library file with those mathematical expression , and corresponding prototype is declared in the header file. In our project we use this lib file and header file for matrix operation. Here we give our own header file and call the function from main().
We used to create .obj file and .lib file for matrix using tcc and tlib command.

II. RELATED WORK

There are many functions in C – Compiler, with direct using of which we can code our required program, like pow(), abs(), sin(), atan(), fabs() etc., but for example, when we code the graphics program, at first some operations of matrix should be coded like matrix multiplications, matrix addition, subtraction, determinant of matrix etc. for shifting, rotating, sharing, scaling etc. But the operation of the matrix is not our actual but the supporting task. In this type of program, programmers are beneficial if they can code only those parts of actual works with the help of in build function. Different operations of Matrix are created in build functions in C – Compiler that can be used in program by directly calling of functions for making easy to code actual coding.

III. ACTUAL WORK

3.1 Matrix Input: In matrix_input, we have taken one pointer type array, one integer for number of row and one integer for number of column. Following function hasbeen written.1st argu ment is use for take input matrix, 2nd argument is use for row of the matrix and 3rd argument is use for column of the matrix

```
void matrix_input(data_type *a[], int r1, int c1)
{
        printf("\nEnter a elements:\n");
        for(i=0;i<r1;i++)
        {
                for(j=0;j<c1;j++)
                {
                        scanf("%d",a+i*c1+j);
                }
        }
}
```

3.2 Matrix Display: In matrix_show, we have taken one pointer type array, one integer for number of row and one integer for number of column. Following function has been written.1st argument is use for display the matrix, 2nd argument is use for row of the matrix and 3rd argument is use for column of the matrix

```
void matrix_show(data_type *a[],int r1,int c1)
{
        printf("Enter the element : \n\n");
        for(i=0;i<r1;i++)
        {
                for(j=0;j<c1;j++)
                {
                        printf("%d\t",*(a+i*c1+j));
                }
                printf("\n\n");
        }
}
```

3.3 Matrix Addition: In matrix addition we have taken three pointer type array , 2 pointer type array for taking input and one for showing output. In the following the function we have written. Here 1st and 2nd argument is use for those matrix between which matrix addition will be done and kept in to the 3rd argument, where as 4th argument and 5th argument are respectively row and column of the matrices.

```
void matrix_add(data_type *a[],data_type *b[],data_type *add1[],int r1,int c1)
{
        for(i=0;i<r1;i++)
        {
                for(j=0;j<c1;j++)
                {
                        data_type t=i*c1+j;
                        data_type ta=*(a+t);
                        data_typetb=*(b+t);
                        *(add1+t)=ta+tb;
                }
        }
}
```

3.4 Matrix Subtraction : In matrix subtraction we have taken three pointer type array , 2 pointer type array for taking input and one for showing output. In the following the function we have written. . In the following the matrix subtraction function we have written.. Here the 3rd argument will be result of subtracting 2nd argument of the 1st, where as 4th argument and 5th argument are respectively row and column of the matrices.

```
void matrix_sub(data_type *a[],data_type *b[],data_type *sub1[],int r1,int c1)
{
        for(i=0;i<r1;i++)
        {
                for(j=0;j<c1;j++)
                {
                        data_type t=i*c1+j;
                        data_type ta=*(a+t);
                        data_typetb=*(b+t);
                        *(sub1+t)=ta-tb;
                }
        }
}
```

3.5 Matrix Multiplication : In matrix multiplication we have taken three pointer type array , 2 pointer type array for taking input and one for showing output. In the following the function we have written. . In the following the matrix multiplication function we have written.. Here 1st and 2nd argument is use for those matrix between which matrix multiplication will be done and kept in to the 3rd argument, where as 4th argument and 5th argument are respectively row and column of the 1st matrix and 6th argument is treated as the numer of columns of 2nd matrix.

```
void matrix_mult(       data_type *a[],data_type *b[],data_type *mul[],intr1,int c1,int c2)
{
        for(i=0;i<r1;i++)
        {
                for(j=0;j<c2;j++)
                {
                        data_typesum=0;
                        for(k=0;k<c1;k++)
                        {
                                data_type t = i * c1+k;
                                data_type p = k * c2+j;
                                data_type ta = *(a+t);
                                data_type pa = *(b+p);
                                sum=sum+ ta * pa;
                        }
                        *(mul+i*c2+j)=sum;
                }
        }
}
```

3.6 Matrix Determinant: In the following we are implemented the function of the matrix determinant. Here function return the determinant value of a matrix, passed as 1st argument having the number of order mentioned in 2nd argument.

```
float matrix_det(data_type *x[],int n)
{
        data_type b[100][100];
        float a[100][100], result=1.0;
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                {
                        b[i][j]=*(x+i*n+j);
                        a[i][j]=b[i][j];
                }
        }
        for(k=0;k<n-1;k++)
        {
                for(j=k+1;j<n;j++)
                {
                        float m=a[j][k]/a[k][k];
                        for(i=0;i<n;i++)
                                a[j][i]=a[j][i]-m*a[k][i];
                }
        }
        for(i=0;i<n;i++)
        {
                result=result*a[i][i];
        }
        return(result);
}
```

3.5 Using command to create .obj file and .lib file :

TCC – We are using tcc command for creating .obj file on the implemented function of the matrix. Step 1. At first we write the code of matrix in notepad and save as it .C extension. Step 2. Then go to the dos prompt and write the command *tcc –c filename.c* Step 3. Create the .obj file for the matrix.

Ex. *tcc –c matrixadd.c*

TLIB- We are using tlib command for creating .lib file of the matrix function. Step1. At first we write the only function for the matrix add ,sum, mul, det because lib file contais only function body. Step2. Then go to the dos prompt and write the command *tlib /C mylib.lib+filename.obj* Step 3. Create the .lib file for the particular function of the matrix.

Ex *.tlib /C mylib.lib + matrixadd.obj*

Header file – We also create the header file for the matrix program. Header file contains only the prototype of the functions that is attached in the main program file before compilation. We write down the only prototype of the function in the notepad and that save into .h extension. In this process we implemented the header file of the particular function. Example: *#include<matrix.h>*

IV. EXAMPLE OF EXECUTION AND RESULT

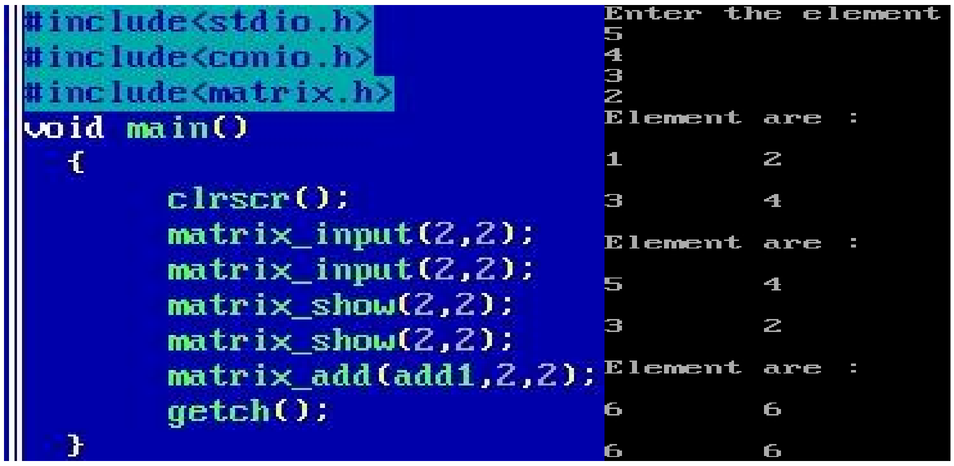
Here we are giving some example of our success fully created library functions. In Black screen (Right Side) of the all figures are shown the output of the executed program which are also shown others screen (Left side) of the corresponding figures. Figure 4.1 shows the example of addition of matrix whereas in Figure 4.2, way of calling functions for matrix subtraction is shown with its successful output of execution of program. Similarly example with successful execution of multiplication between two matrixes is displayed in figure 4.3. The determinant of a matrix is being focused in figure 4.4. All though, in all the cases, taking the input of matrix and displaying the matrix is respectively done through calling of functions. matrix_input() and matrix_show().

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

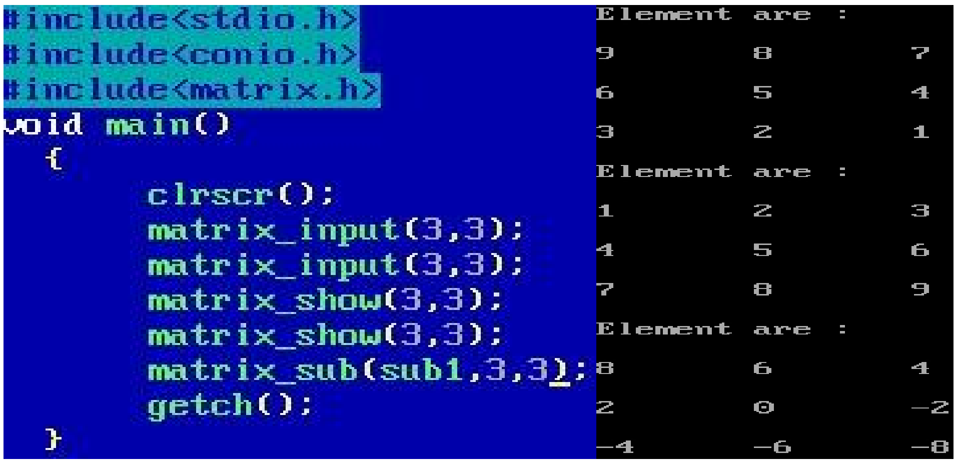
Vol. 4, Issue 5, May 2016



```
#include<stdio.h>
#include<conio.h>
#include<matrix.h>
void main()
  {
        clrscr();
        matrix_input(2,2);
        matrix_input(2,2);
        matrix_show(2,2);
        matrix_show(2,2);
        matrix_add(add1,2,2);
        getch();
  }
```

```
Enter the element
5
4
3
2
Element are :
1       2
3       4
Element are :
5       4
3       2
Element are :
6       6
6       6
```

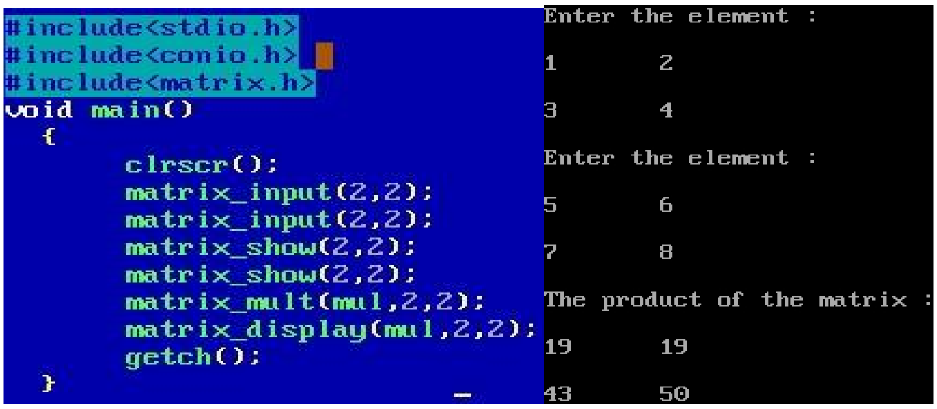
Figure 4.1
Coding for calling of matrix addition and output



```
#include<stdio.h>
#include<conio.h>
#include<matrix.h>
void main()
  {
        clrscr();
        matrix_input(3,3);
        matrix_input(3,3);
        matrix_show(3,3);
        matrix_show(3,3);
        matrix_sub(sub1,3,3);
        getch();
  }
```

```
Element are :
9       8       7
6       5       4
3       2       1
Element are :
1       2       3
4       5       6
7       8       9
Element are :
8       6       4
2       0       -2
-4      -6      -8
```

Figure 4.2
Coding for calling of matrix subtraction and output



```
#include<stdio.h>
#include<conio.h>
#include<matrix.h>
void main()
  {
        clrscr();
        matrix_input(2,2);
        matrix_input(2,2);
        matrix_show(2,2);
        matrix_show(2,2);
        matrix_mult(mul,2,2);
        matrix_display(mul,2,2);
        getch();
  }
```

```
Enter the element :
1       2
3       4
Enter the element :
5       6
7       8
The product of the matrix :
19      19
43      50
```

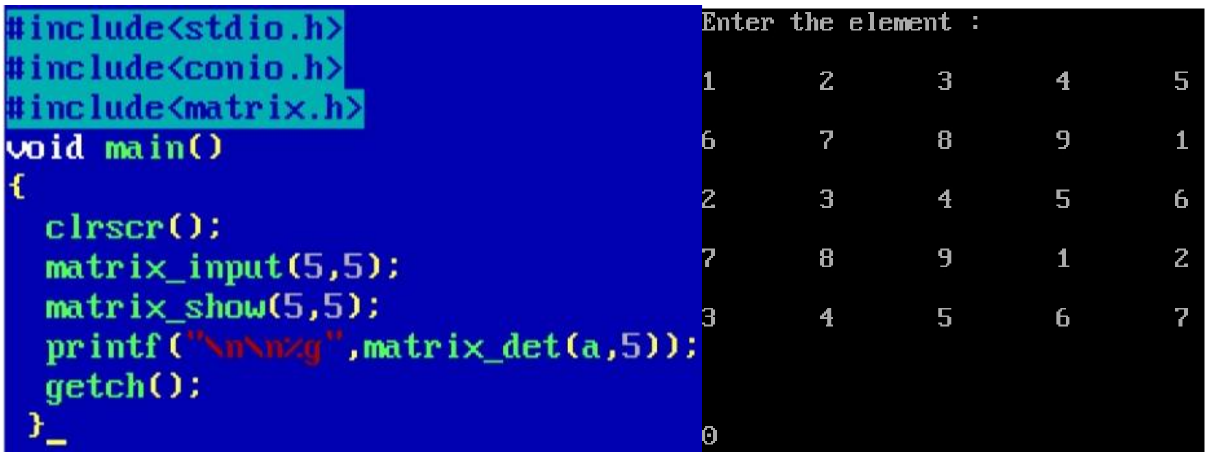
Figure 4.3
Coding for calling of matrix multiplication and output

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016



```
#include<stdio.h>
#include<conio.h>
#include<matrix.h>
void main()
{
  clrscr();
  matrix_input(5,5);
  matrix_show(5,5);
  printf("\n\n%g",matrix_det(a,5));
  getch();
 }_
```

```
Enter the element :

1       2       3       4       5

6       7       8       9       1

2       3       4       5       6

7       8       9       1       2

3       4       5       6       7


0
```

Figure 4.4
Coding for calling of matrix determinant and output

V. ANALYSIS

When this matrix functions are used by any other programmer, any order of matrix may be defined in the code of the program. When this matrix is being passed into the our proposed library functions, like matrix_add(), matrix_mult() etc. Basically matrix is 2 dimensional array. To pass a 2D array in a function,size of any one dimension of the array may be varied. Size of the other damnations should be given, like a[10][] or a[][10]. But in that case, if we pass such manner, the functions in which matrix is being passed as 2D array, cannot be work with proper order of the matrix. 2D array is an array of 1D array, which can be passed into the functions without mentioning the size of 1D array. Array is pointer. If pointer of 1D Array can be passed, no size of the array will be mentioned. To pass 2D array, without mentioning the size, *a[] can be passed for any order of the matrix.
Here we use pointer type array because we hold the address value of the array and then take the input value into array. Here we use this function into library file create a.exe file. We take pointer type array to hold the address value of the pointer. Arrays are very useful but C does not have array variables. This might seem incorrect as there is a C syntax for working with arrays. For example one can create an array, unimaginatively called 'Array1', of 9 integers by int Array1[9];, store the number '96' in the 8th element (elements being numbered from 0 not from 1) withArray1[7]=96;and print it that element with printf("%d", Array1[7]);but this is really just working with pointers with an alternative syntax. Therefore the lineArray1[7]=96;is identical to*(Array1+7)=96; because 'Array1' is just a pointer to the start of the memory chunk and the '[7]' moves it along 7 positions then uses what is at that position there. A particularly confusing special case is that "*Array1" can be used, of course, as a short way of typing "Array1[0]" but looks nothing like an array at all In C one can budge up an such an array by bypassing the array allocation line, explicitly creating a pointer and a chunk of memory of the correct size and using that instead. For example, int *Array1;so we take address value of the array and store the array value in that address , and pass through parameter. and we done our operation by holding our address position.

VI. CONCLUSION

In this enhanced compiler, it is clearly nodded that without writing the coded for following the operations matrix, programmer can code by only calling of the functions of those operations.Function – prototypes of all these functions have been written in a header file, named matrix.h which is only needed to include for calling the these functions.

1. Input of the Matrix	:	by the function, matrix_input()
2. Display of the Matrix	:	by the function,matrix_show()
3. Addition between two Matrices	:	by the function,matrix_add()
4. Subtraction 2nd Matrix from 1st Matrix	:	by the function,matrix_sub()
5. Multiplication between two Matrices	:	by the function,matrix_mult()

6. Determinant of Matrix : by the function,matrix_det()

REFERENCES

[1] C: THE COMPLETE REFERENCE (4TH ED) , AUTHOR : HERBERT SCHILDT.
[2] C Programming Language (2nd Edition By B. W. Kernighan & D. M. Ritchie)
[3] C Programming: A Modern Approach, 2nd Edition By K. N. King
[4] Prof. (Dr.) PranamPaul,“Implementation of InformationSecurity based on Common Division”, International Journal of Computer Science and Network Security, Vol.11, No. 2,2011.
[5] Data Structures Using C - Aaron M. Tenenbaum
[6] C Programming in 12 Easy Lessons by Greg Perry
[7] John C. Bowman,"Math 422 Coding Theory & Cryptography", University of Alberta, Edmonton, Canada
[8] C: A Reference Manual by Samuel P. Harbison and Guy R. Steele
[9] C Programming: A Modern Approach by K. N. King
[10] Learn C The Hard Way by Zed Shaw
[11] Prof. (Dr.) PranamPaul,“An Application to ensure Security through Bit-level Encryption”, International Journal of Computer Science and Network Security, Vol. 9, No. 11, 2009.
[12] The C Book by Mike Banahan, Declan Brady and Mark Doran
[13] The Standard C Library by P.J. Plauger
[14] Pranam Paul, SaurabhDutta, A K Bhattacharjee,”AnApproach to ensure Security through Bit-level Encryption withPossible Lossless Compression”, International Journal ofComputer Science and Network Security”, Vol. 08, No. 2, pp.291 – 299,2008
[15] Algorithms in C by Robert Sedgewick
[16] SukanyaChakravarty, Prof. Dr. Pramnam Paul,”Approach Based on Finding the Difference between Consecutive Numbers”, International Journal of Innovative Research in Computer and Communication Engineering, ISSN(Online): 2320-9801,ISSN (Print): 2320-9798 vol-4,Issue-2,2016.
[17] Pointers on C by Kenneth Reek
[18] https://en.wikipedia.org/wiki/C_%28programming_language%29
[19] http://www.programiz.com/c-programming/examples/matrix-multiplication-function
[20] http://www.programmingsimplified.com/c-program-matrix

BIOGRAPHY

Pritam Ghosh is a Final year Student of MCA in Narula Institute of Technoloy , Agarpara, WestBengal India. He completed his BCA degree from George College (Kolkata) under the WBUT.

Sudip Sinha is a Final year Student of MCA in Narula Institute of Technoloy , Agarpara, WestBengal India.
He completed his Bsc degree from Bhairab Ganguly College (kolkata) under the WBSU .

Prof. Dr Pranam Paul, Assistant Professor and Departmental Head, CA Department, Narula Institute of Technology (NIT), Agarpara had completed MCA in 2005. Then my carrier had been started as an academician from MCKV Institute of Technology, Liluah. Parallely, at the same time, I continued my research work. At October, 2006, National Institute of Technology (NIT), Durgapur had agreed to enroll my name as a registered Ph.D. scholar. Then I had joined Bengal College of Engineering and Technology, Durgapur. After that Dr. B. C. Roy Engineering College hired me in the MCA department at 2007. At the age of 30, I had got Ph.D. from National Institute of Technology, Durgapur, WestBengal. I had submitted his Ph.D. thesis only within 2 Years and 5 Months. After completing the Ph.D., I had joined Narula Institute of Technology in Computer Application Department. Parallely I continue my research work. For that, I have 39 International Journal Publications among 54 accepted papers in different areas. I also reviewer of International Journal of Network Security (IJNS), Taiwan and International Journal of Computer Science Issue (IJCSI); Republic of Mauritius.

Achievements:

1. Selected his name as “Top 100 Engineers’ 2011”, by “International Biographical Centre”, Cambridge, England
2. Selected his name as “Outstanding 2000 Intellectuals of the 21st Century, 2012”, by “International Biographical Centre”, Cambridge, England