|  |  |  |
| --- | --- | --- |
| **Mykyev A. Bakirovich**  FYQWAV  [fyqwav@inf.elte.hu](mailto:fyqwav@inf.elte.hu)  Group 4 | 1. **Assignment/2. task** | 19th February 2022 |

**Task**

*Implement the X matrix type which contains integers. These are square matrices that can contain nonzero entries only in their two diagonals. Don't store the zero entries. Store only the entries that can be nonzero in a sequence. Implement as methods: getting the entry located at index (i, j), adding and multiplying two matrices, and printing the matrix (in a square shape).*

**X matrix type**

***Set of values***

*X*(n) =

***Operations***

1. *Getting an entry*

Getting the entry of the ith column and jth row

Formally:

This operation needs any action only if i=j or if i+j=n+1, otherwise output is zero.

1. *Addition*

Addition of two matrices: c = a + b. The matrices have the same size.

Formally:

In case of X matrices there is an easier version:

1. *Multiplication*

Multiplication of two matrices: c = a\*b. The matrices have the same size.

Formally:

In case of X matrices there is an easier version:

1. Printing the matrix

Printing the matrix a with the help of a console out stream (cout).

Formally:

In case of X matrices there is an easier version:

***Representation***

Only the X diagonal of the n n matrix has to be stored.

Only a one-dimension array () is needed, with the help of which any entry of the X matrix can be get:

***Implementation***

1. Getting an entry

Getting the entry of the ith column and jth row where the matrix is represented by and n stands for the size of the matrix can be implemented as

1. *Addition*

The result of addition of matrices a and b (represented by arrays t and ) goes to matrix c (represented by array u), where all of the array have to have the same size.

1. Multiplication

The result of multiplication of matrices a and b (represented by arrays t and ) goes to matrix c (represented by array u), where all of the array have to have the same size.

1. Printing the matrix

Printing the matrix a where it is represented by array , and n stands for the size of the matrix can be implemented as:

***Testing***

Testing the operations (black box testing)

1. Creating, reading, and writing matrices of different size.
2. 1, 2, 5-size matrix
3. Setting an entry
4. Setting entry in the beginning and at the end in the diagonal
5. Setting entry in the beginning and at the end out of the diagonal
6. Illegal index, indexing a 0-size matrix
7. Copy constructor
8. Creating a matrix b based on matrix a, comparing the entries of the two matrices. Then, changing one of the matrices and comparing the entries of the two matrices.
9. Assignment operator
10. Executing b=a for matrices a and b (with and without same size), comparing the entries of the two matrices. Then, changing one of the matrices and comparing the entries of two matrices.
11. Executing b=a=c for matrices a and b and c (with and without same size), comparing the entries of the three matrices. Then, changing one of the matrices and comparing the entries of three matrices.
12. Executing a=a for matrix a, comparing its entries with the original matrix which is pre-copied to the matrix temp and comparing the entries of two matrices.
13. Addition of two matrices, command c: = a+b.
14. With matrices of different size (size of a and b differs, size of c and a differs)
15. Checking the commutativity (a + b == b + a)
16. Checking the associativity (a + b + c == (a + b) + c == a + (b + c)
17. Checking the neutral element (a + 0 == a, where 0 is the null matrix)
18. Multiplication of two matrices, command c := a\*b.
19. With matrices of different size (size of *a* and *b* differs, size of *c* and *a* differs)
20. Checking the commutativity (a \* b != b \* a)
21. Checking the associativity (a \* b \* c == a \* (b \* c) == (a \* b) \* c
22. Checking the neutral element (a \* 0 == 0, where 0 is the null matrix)
23. Checking the identity element (a \* 1 == a, where 1 is the identity matrix)

Testing based on the code (white box testing)

1. Creating an extreme-size matrix (-1,1,1000).
2. Generating and catching exceptions