UDC 004.422.632

APPLICATION OF MATRIX IN PROGRAMMING

Grenberg O.K.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Sokolova M. A. – Lecturer at the Department of Foreign Languages

Annotation: This article explores the basic principles of working with matrices in programming, their application in various fields and methods for optimizing operations with them. From simple arithmetic operations to complex machine learning algorithms, matrices are the basis of many modern computing technologies.

Keywords: matrix, programming, information, 2D Visualization, algorithm, technology, robotics, data, simulation, geometric transformations.

Introduction. Matrices play a crucial role in modern programming, serving as a fundamental tool across various fields. These two-dimensional data structures enable efficient storage and manipulation of data organized in rows and columns. By using matrices, developers can tackle complex problems that involve large datasets and intricate calculations, streamlining processes for both developers and end users.

Main part. Before we embark on this exploration, let us first illuminate the concept of a matrix. In the realm of mathematics, a matrix is a structured tableau, a rectangular array meticulously arranged with numbers, symbols, or expressions. These elements, like players on a field, occupy specific positions defined by rows and columns. This arrangement serves as a powerful tool, a representation of a mathematical object or one of its inherent properties.

In the figure 1 there is a matrix gracefully composed of a three row and three columns. We often refer to it as a «three-by-three matrix,» a concise «matrix,» or a matrix of a specific dimension. Now, let us unfurl a panorama of ten distinct matrix archetypes: the Row Matrix, the Column Matrix, the Null Matrix, the Square Matrix, the Diagonal Matrix, the Sparse Matrix, the Spiral Matrix, the Identity Matrix, the Triangular Matrix, and the Symmetric Matrix [1, 4].

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 4 \\ 3 & 2 & 3 \end{pmatrix}$$

Figure 1 – An example of a matrix

Matrices are fundamental in programming across numerous fields, though we'll focus on key applications. In computer graphics, they enable precise object transformations like rotations and scaling, control projections and camera motion. For image processing, matrices handle pixel manipulations through convolutions. Robotics relies on matrices to choreograph limb movements in multi-dimensional spaces. Artificial intelligence uses matrices to manage and prepare large datasets for machine learning. Game developers employ matrices for collision detection and transforming object properties. Finally, in game physics engines, matrices govern the positions, rotations, and movement of objects based on applied forces [2, 3].

There’s an example of using a matrix to find the cell number of the maximum number in C++ (shown in Figure 2):

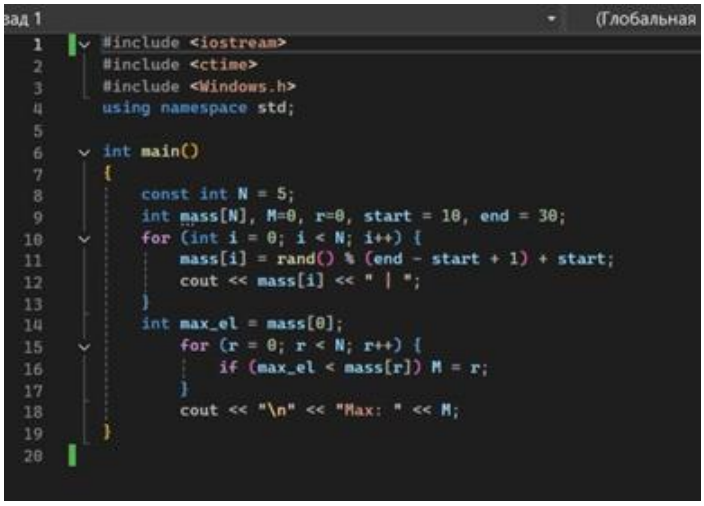

Figure 2 – A matrix of finding the cell number of the maximum number in C++

Having glimpsed the pervasive applications of matrices, let us now consider the shimmering advantages and stark disadvantages that accompany their use. Among their strengths, matrices offer a compelling visual aid in 2D representation, elegantly consolidating multiple elements of identical type under a single, unifying name. Like keys to a treasure chest, indices grant swift, direct access to individual elements, allowing for truly random retrieval. Matrices stand ready to house data of all forms, provided it adheres to the immutable law of fixed size. Simplicity graces their very essence, making them readily implementable, the quintessential 2D data structure embraced by the vast lexicon of programming languages.

However, the rigid nature of matrices presents inherent limitations. Their dimensions must be predetermined, an immutable constraint. This fixed size can become a significant drawback, as matrices offer no flexibility for dynamic resizing. Insertion and deletion operations can be costly affairs, demanding a cumbersome shift of elements to accommodate the change. Furthermore, memory wastage looms as a potential issue, arising when the matrix is allocated with more space than ultimately required. The very act of resizing can be a time-consuming ordeal, particularly when dealing with frequently resized or exceptionally large matrices. Finally, matrices often lack built-in functionalities, leaving developers to implement common operations from scratch [2, 4].

Conclusions. In conclusion, matrices stand as indispensable instruments in the programmer's arsenal, offering an elegant and powerful means to structure and manipulate multidimensional data. Their influence permeates diverse technological landscapes, from the captivating realms of computer graphics and the intricate algorithms of machine learning to the nuanced art of image processing, the rigorous demands of scientific computing, and the complex webs of network analysis. Matrices empower us to orchestrate complex transformations, encode rich data representations, and unravel intricate mathematical puzzles, such as systems of equations. Within the domain of machine learning, they serve as the very bedrock upon which we train intelligent models, while in the visual splendor of computer graphics, they breathe life into objects through seamless transformations. Their inherent versatility, coupled with their computational efficiency, solidifies their position as an indispensable cornerstone across the vast expanse of programming and technology.

References

1. *The matrix, its history and application [Electronic resource]. Mode of access: https://urok.1sept.ru/articles/637896. Date of access: 27.02.2025.*

2. *Matrix Data Structure [Electronic resource]. Mode of access: https://medium.com/@Mandeep2002/matrix-data-structure-f06246c52ead. Date of access: 02.03.2025.*

3. *Real life application of Matrices [Electronic resource]. Mode of access: https://www.geeksforgeeks.org/real-life-application-of-matrices/. Date of access: 05.03.2025.*

4. *Applications, Advantages and Disadvantages of Matrix Data Structure [Electronic resource]. Mode of access: https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-matrix-data-structure/#. Date of access: 10.03.2025.*