At the beginning of the program, write a compiler directive \texttt{#include "bibs.h"}.

1. Define the function \texttt{void abx(double** a, double* b, double* x, int n)} implementing the \textit{Gaussian Elimination Algorithm} for the system of linear equations

\[
a x = b
\]  

where \(a \in M_{n\times n}\) is the known, nonsingular matrix, of the \textit{rank} \(a = n\), \(b \in \mathbb{R}^n\) is the known vector, and \(x \in \mathbb{R}^n\) represents the vector of the unknown solution of (1).

\textit{Gaussian Elimination Algorithm} – the outline of the method

We assume that \(a_{0,0} \neq 0\), and modify the rows of the matrix \(a \in M_{n\times n}\) and the elements of the vector \(b \in \mathbb{R}^n\) with indices \(k = 1, 2, \ldots, n-1\) calculating: \(a_{k,j}^1 = a_{k,j} - \frac{a_{k,0}}{a_{0,0}} a_{0,j}, b_k^1 = b_k - \frac{a_{k,0}}{a_{0,0}} b_0\) for \(j = k-1, k, \ldots, n-1\), i.e. the matrix \(a\) and vector \(b\)

\[
\begin{bmatrix}
a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\
a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1}
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1 \\
\vdots \\
b_{n-1}
\end{bmatrix}
\]

are transformed to the following matrix \(a^1\) and vector \(b^1\)

\[
\begin{bmatrix}
a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\
a_{1,0} & a_{1,0} a_{0,0} & a_{1,1} & \cdots & a_{1,n-1} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n-1,0} & a_{n-1,0} a_{0,0} & a_{n-1,1} & \cdots & a_{n-1,n-1}
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1 - \frac{a_{1,0}}{a_{0,0}} b_0 \\
\vdots \\
b_{n-1} - \frac{a_{n-1,0}}{a_{0,0}} b_0
\end{bmatrix}
\]

The new and modified system of linear equations (but equivalent to the system (1))

\[
a^1 x = b^1
\]  

is defined by the matrix \(a^1 \in M_{n\times n}\), with the all elements in the first column and below the element \(a_{0,0}\) equal to 0:
The above transformations should be repeated for sub-matrix and sub vector shown below:

\[
\begin{bmatrix}
... & a_{1,1} & 0 & ... & 0 \\
... & a_{2,1} & a_{1,2} & ... & a_{2,n-1} \\
... & ... & ... & ... & ... \\
... & a_{n-1,1} & a_{n-1,2} & ... & a_{n-1,n-1}
\end{bmatrix}
\begin{bmatrix}
... \\
b_1 \\
... \\
b_{n-1}
\end{bmatrix}
\]

Now we assume that \(a_{1,1} \neq 0\), and modify the rows of the matrix \(a^1 \in M_{n \times n}\) and the elements of the vector \(b^1 \in \mathbb{R}^n\) with indices \(k = 2, 3, ..., n-1\) calculating:

\[a_{k,j}^2 = a_{k,j}^1 - a_{k,1}^1 a_{1,j}^1 / a_{1,1}^1, \quad b_k^2 = b_k^1 - a_{k,1}^1 b_1^1 / a_{1,1}^1 \quad \text{for} \quad j = k-1, k, ..., n-1\]

We get the modified system of linear equations

\[a^2 x = b^2\]
If we repeat the described above procedure many times, we will modify the initial system \( a \times b = \) to
\[
(a^{-1})n \times n = a \times b 
\] (with upper triangular matrix), where

\[
\begin{bmatrix}
a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,n-1} \\
0 & a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} \\
0 & 0 & a_{2,2} & \cdots & a_{2,n-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & a_{n-1,n-1}
\end{bmatrix}
= \begin{bmatrix}
b_0 \\
b_1 \\
b_2 \\
\vdots \\
b_{n-1}
\end{bmatrix} = b^{-1} 
\] (11)

In the case when some of the elements \( a_{0,0}, a_{1,1}, a_{2,2}, \ldots \) are equal to 0, it is necessary to find (in the simplest variant) the first row \( k \) (“below the row \( i-1 \)”) in which the element \( a_{k,0} (k > 0), a_{k,1} (k > 1), a_{k,2} (k > 2), \ldots \) is not equal to 0 and exchange the row with the zero element with the found row \( k \). In practice, the finding of the appropriate elements is more complicated and is the result of the minimization of the round-off and truncation numerical errors that occur during the described calculations.

If we denote the matrix \( a^{-1} \in M_{n \times n}, a_{i,j}^{-1} (i, j = 0,1,\ldots,n-1) \), and vector \( b^{-1} \in \mathbb{R}^n, b_i^{-1} (i = 0,1,\ldots,n-1) \) as \( a = (a_{i,j}) \in M_{n \times n} \) and \( b = (b_i) \in \mathbb{R}^n \), respectively, we can say that the problem of finding the solution of the system of linear equations (1) is transformed to the equivalent problem
\[
a \times b = b 
\] (12)

with upper triangular matrix \( a \in M_{n \times n} \), where

\[
a = \begin{bmatrix}
a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,n-1} \\
0 & a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} \\
0 & 0 & a_{2,2} & \cdots & a_{2,n-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & a_{n-1,n-1}
\end{bmatrix} \in M_{n \times n}, \quad b = \begin{bmatrix}
b_0 \\
b_1 \\
b_2 \\
\vdots \\
b_{n-1}
\end{bmatrix} \in \mathbb{R}^n. 
\] (13)

The solution \( x = (x_j) \in \mathbb{R}^n \) of the system (12) can be very easily found implementing the back substitution loop over the result from the forward pass (upper triangular matrix), working from the last equation to first as follows:

\[
x_{n-1} = \frac{b_{n-1}}{a_{n-1,n-1}},
\]

\[
x_{i} = \frac{b_i - \sum_{j=i+1}^{n-1} a_{i,j} x_j}{a_{i,i}} \quad (i = n-2, n-3, \ldots, 1, 0).
\]
Gaussian Elimination Algorithm (simplest version) – pseudo-code

```plaintext
start
for each row \( i = 1, 2, ..., n - 1 \)
{
  if \( a_{i-1,j-1} = 0 \), then find the first row \( k \) \((i \leq k \leq n - 1)\), such that \( a_{k,j-1} \neq 0 \), and next replace (mutually) the row \( i - 1 \) with the row \( k \) of the matrix \( a \). Similarly, replace (mutually) the element \( i - 1 \) with the element \( k \) of the vector \( b \), i.e.
  \( a_{i-1,j} \leftrightarrow a_{k,j} \ (j = i - 1, i, ..., n - 1), \quad b_{i-1} \leftrightarrow b_k \)
  (row changing can be limited only to the elements from \( i - 1 \) column of the matrix \( a \))
  for each row \( k \) \((i \leq k \leq n - 1)\)
  { 
    \( c = \frac{a_{k,j}}{a_{i-1,j-1}} \)
    \( a_{k,j-1} = 0 \)
    for each column \( j \) \((i \leq j \leq n - 1)\)
    { 
      \( a_{k,j} = a_{k,j} - c \ a_{i-1,j} \)
    }
    \( b_k = b_k - c \ b_{i-1} \)
  }

\( x_{n-1} = \frac{b_{n-1}}{a_{n-1,n-1}} \)
for each element \( i = n - 2, n - 3, ..., 1, 0 \)
{
  \( b_j - \sum_{j=i+1}^{n-1} a_{i,j} \ x_j \)
  \( x_i = \frac{b_j}{a_{i,i}} \)
}
end

2. Test the function \( \text{void} \ abx(\text{double**} \ a, \text{double*} \ b, \text{double*} \ x, \text{int} \ n) \) for the matrix \( A \in M_{N \times N} \) and vector \( B \in \mathbb{R}^N \) with randomly generated elements from the interval \([-10, 10]\) for \( N = 10 \). Display the solution \( X \in \mathbb{R}^N \) of the system of linear equations \( A \ X = B \), matrix \( A \in M_{N \times N} \) and vector \( B \in \mathbb{R}^N \) before and after calling the function \( \text{void} \ abx(...) \).

3. Call the function \( \text{int} \ \text{gauss(} \text{int} \ n, \text{double**} \ a, \text{double*} \ b, \text{double*} \ x) \) from the library \( \text{bibs.h} \), for the matrix \( A \in M_{N \times N} \) and vector \( B \in \mathbb{R}^N \), from Ex.2. Compare the both results \( X \in \mathbb{R}^N \).
```
4. Test the function `void abx(double ** a, double * b, double * x, int n)` for the following matrix \( A \in M_{2 \times 2} \) and vector \( B \in \mathbb{R}^2 \)\( (N = 2) \)

\[
A = \begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \in M_{2 \times 2}, \quad B = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \in \mathbb{R}^2, \tag{14}
\]

where \( \varepsilon \) is the „very small” positive number, e.g.: \( \varepsilon = 1.0 \cdot 10^{-20} \).

**Commentary.** The exact solution \( X = \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} \in \mathbb{R}^2 \) of the system of equations

\[
A \ X = B \tag{15}
\]

i.e.

\[
\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{16}
\]

is the following

\[
X = \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} = \begin{bmatrix} 1 - \frac{1}{\varepsilon} \\ \frac{2 - \frac{1}{\varepsilon}}{\varepsilon} \\ \frac{1 - \frac{1}{\varepsilon}}{\varepsilon} \\ \frac{1 - \frac{1}{\varepsilon}}{1 - \frac{1}{\varepsilon}} \end{bmatrix} = \ldots
\]

Gaussian Elimination Algorithm (in the simplest version) transforms the system of linear equations (16) to the following one:

\[
\begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - \frac{1}{\varepsilon} \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{2 - \frac{1}{\varepsilon}}{\varepsilon} \end{bmatrix}, \tag{18}
\]

that has the following exact solution

\[
X_1 = \frac{2 - \frac{1}{\varepsilon}}{1 - \frac{1}{\varepsilon}}, \quad X_0 = \frac{1 - X_1}{\varepsilon}. \tag{19}
\]

**Very Important !** The order \( X_1, X_0 \) in (19) is the same as the order of calculated components \( X_1, X_0 \) in Gaussian Elimination Algorithm – see above.
For very small $\varepsilon > 0$, e.g. $\varepsilon = 1.0 \cdot 10^{-20}$, the values of the both expressions $2 - \frac{1}{\varepsilon}$ and $1 - \frac{1}{\varepsilon}$ in (19) are nearly the same and are approximately equal to $\frac{1}{\varepsilon}$, i.e. $2 - \frac{1}{\varepsilon} \approx -\frac{1}{\varepsilon}$ and $1 - \frac{1}{\varepsilon} \approx -\frac{1}{\varepsilon}$. So, from the formula (19) we get the following numerical result

$$X_1 = \frac{2 - \frac{1}{\varepsilon}}{1 - \frac{1}{\varepsilon}} \approx \frac{-\frac{1}{\varepsilon}}{-\frac{1}{\varepsilon}} = 1,$$

(20)

and consequently, from the formula (19) we calculate

$$X_0 = \frac{1 - X_1}{\varepsilon} \approx \frac{1 - 1}{\varepsilon} = 0.$$

(21)

But, the simple transformation of the formulae (19) (compare also with (17)) gives the correct numerical result:

$$X_1 = \frac{1 - 2\varepsilon}{1 - \varepsilon} \approx 1, \quad X_0 = \frac{1}{1 - \varepsilon} \approx 1.$$

(22)

We conclude that the numerical solution found by the simplest version of the Gaussian Elimination Algorithm is not acceptable in this case. The loss of precision is caused by accumulated round-off errors in the numerical calculations. On our example, the element $a_{0,0} = \varepsilon$ is very small in compare with the element $a_{0,1} = 1$. We can minimize round-off problems by rearranging the equations such that the largest coefficient in each equation is placed on the principal diagonal of the coefficient matrix, using one of the permissible operations that allows equations to be rearranged without changing the solution. Many other manipulations are also permissible.

Let us see, that the simple rearranging of the equations (16)

$$\begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

(23)

and applying next the simplest version of the Gauss Elimination Algorithm, results in the equivalent system of equations

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 - \varepsilon \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 - 2\varepsilon \end{bmatrix},$$

(24)

that (this time) has the correct numerical solution.
\[ X_i = \frac{1 - 2 \varepsilon}{1 - \varepsilon} \approx 1 \]  

(25)

and

\[ X_0 = 2 - x_i \approx 1. \]  

(26)

5. Once more test the function `void abx(...)`, for the following defined matrix \( \mathbf{A} \in M_{2 \times 2} \) and vector \( \mathbf{B} \in \mathbb{R}^2 \) \((N = 2)\)

\[
\mathbf{A} = \begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix} \in M_{2 \times 2}, \quad \mathbf{B} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \in \mathbb{R}^2
\]

(27)

(simple rearranging of the equations in (16)). Display the numerical solution \( \mathbf{X} \in \mathbb{R}^N \).

Similarly, for the defined in (27) matrix \( \mathbf{A} \in M_{N \times N} \) and vector \( \mathbf{B} \in \mathbb{R}^N \), call once more the function `int gauss(int n, double** a, double* b, double* x)` from the library `bibs.h` and display the numerical solution \( \mathbf{X} \in \mathbb{R}^N \). Compare the both results with the non-correct numerical solution (20), (21) and with the correct numerical solution (25), (26).