Лабораторна робота 11. Задачі лінійної алгебри

Зміст заняття: Робота з одновимірними і двовимірними масивами. Множення матриць. Транспонування матриць. Розв'язування системи лінійних рівнянь методом Гауса.

1. Знаходження добутку матриць

Добутком $A \cdot B$ двох матриць буде матриця C, елементи якої $C_{i,j}$ рівні сумі попарних добутків елементів рядка першої матриці A на відповідні елементи стовпця другої матриці B:

$$C_{i,j} = \sum_{k=1}^{n} A_{i,k} \cdot B_{k,j}$$

Наприклад, другий елемент першого рядка обчислюється як сума попарних добутків відповідних елементів першого рядка і другого стовпчика.

З цього слідує, що перемножити між собою можна матриці в яких кількість стовпців першої A рівна кількості рядків другої B. Нова матриця C, яка є добутком двох, має розмірність m*n, де m – кількість рядків першої матриці, а n – стовпців другої.

Приклад 11.1. Ввести дві цілочисельні матриці 3×3 і обчислити добуток цих матриць.

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 4 & 0 \\ 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 0 & 1 \\ -1 & 1 & 0 \\ 2 & 5 & 7 \end{bmatrix} =$$

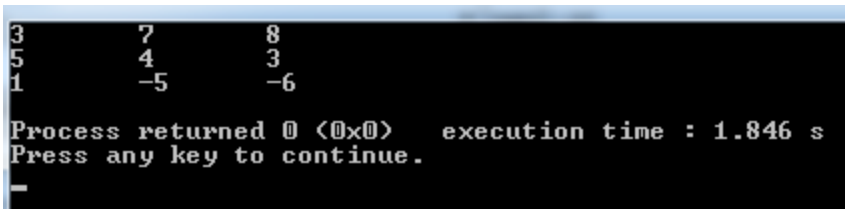
$$= \begin{bmatrix} 1\cdot 3+2\cdot(-1)+1\cdot 2 & 1\cdot 0+2\cdot 1+1\cdot 5 & 1\cdot 1+2\cdot 0+1\cdot 7 \\ 3\cdot 3+4\cdot(-1)+0\cdot 2 & 3\cdot 0+4\cdot 1+0\cdot 5 & 3\cdot 1+4\cdot 0+0\cdot 7 \\ 1\cdot 3+0\cdot(-1)+-1\cdot 2 & 1\cdot 0+0\cdot 1+-1\cdot 5 & 1\cdot 1+0\cdot 0+-1\cdot 7 \end{bmatrix} = \begin{bmatrix} 3 & 7 & 8 \\ 5 & 4 & 3 \\ 1 & -5 & -6 \end{bmatrix}$$

Програмна реалізація:

```
#include <iostream>
using namespace std;
int main()
{
    const int N = 3; // розмірності матриць
    int sum;  // змінна для суми попарних добутків відповідних
елементів матриць A та B

    // ініціалізація матриць
    int A[N][N] = {{1, 2, 1}, {3, 4, 0},{1, 0 ,-1}};
    int B[N][N] = {{3, 0, 1}, {-1, 1, 0},{2, 5 ,7}};
    int C[N][N];

    // множення матриць
    for (int i = 0; i < N; ++i)
    {
        for (int j = 0; j < N; ++j)
        {   sum = 0;
            for (int k = 0; k < N; ++k)
                sum = sum + A[i][k] * B[k][j];
            C[i][j] = sum;
        }
    }
```

```
    // виведення матриці C на екран
    for(int i = 0; i < N; ++i)
    {       for(int j = 0; j < N; ++j)
                cout << C[i][j] << "\t";
          cout << endl;
    }

    return 0;
}
```

Результат виконання програми:



2. Транспонування матриць

Транспонування матриці – це операція над матрицею, при якій її рядки та стовпці міняються місцями:

$$A_{ij}^T = A_{ji}$$

Якщо матриця A має розмір $n \times m$, то транспонована матриця A^T має розмір $m \times n$.

Приклад 11.2. Знайти транспоновану матрицю A^T для матриці

$$A = \begin{pmatrix} 2 & -3 & 4 \\ 1 & 0 & -1 \end{pmatrix}.$$

Розв'язок:

$$A^T = \begin{pmatrix} 2 & 1 \\ -3 & 0 \\ 4 & -1 \end{pmatrix}$$

Програмна реалізація:

```
#include <iostream>
using namespace std;
int main()
{
    const int N = 2, M = 3; // розмірності матриць

    // ініціалізація матриць
    int A[N][M] = {{2, -3, 4}, {1, 0, -1}};
    int AT[M][N];

    // Транспонування матриці
    for(int i = 0; i < M; ++i)
    {
        for(int j = 0; j < N; ++j)
        {
            AT[i][j] = A[j][i];
```

```
        }
    }

    // виведення матриці AT на екран
    for(int i = 0; i < M; ++i)
    {       for(int j = 0; j < N; ++j)
                cout << AT[i][j] << "\t";
          cout << endl;
    }

    return 0;
}
```

Результат виконання програми:

```
2       1
-3      0
4       -1

Process returned 0 (0x0)   execution time : 0.281 s
Press any key to continue.
```

3. Розв'язання систем лінійних рівнянь (СЛАР)

Багато проблем у різних галузях науки й техніки призводять до розв'язання однієї й тієї ж математичної задачі – розв'язання системи лінійних алгебраїчних рівнянь. Розглянемо для прикладу задачу визначання струму в електричному колі, подану на рис. 11.1.

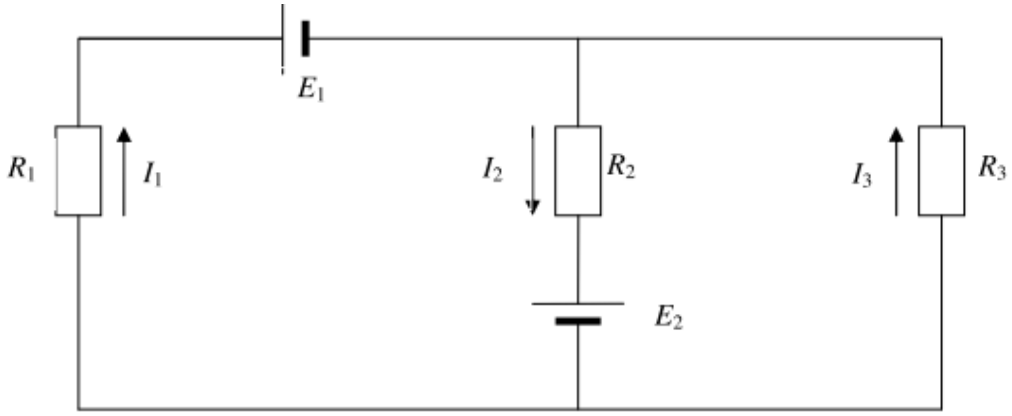


Рис. 11.1. Схема електричного кола.

Опори R_1 , R_2 , R_3 ділянок кола й електрорушійні сили E_1 і E_2 джерел струму (ЕРС) є відомими. Додатні напрямки струмів показано на рисунку стрілками. Відповідно до першого закону Кірхгофа, алгебраїчна сума струмів, що сходяться у вузлі, дорівнює нулеві. Другий закон Кірхгофа стверджує, що в кожному замкнутому контурі електричного кола алгебраїчна сума добутків сил струмів I_i на опори R_i відповідних ділянок цього контура дорівнює алгебраїчній сумі прикладених до нього ЕРС (i = 1, 2, …, n). Використовуючи ці закони для кола, зображеного на рис. 11.1, можна записати:

$$\begin{aligned} -I_1 + I_2 - I_3 &= 0; \\ R_1 I_1 + R_2 I_2 &= E_1 + E_2; \\ R_2 I_2 + R_3 I_3 &= E_2. \end{aligned}$$

Отже, маємо розв'язати систему лінійних алгебраїчних рівнянь з невідомими I_1, I_2, I_3. Розв'язки цієї системи рівнянь – це значення сил струмів I_i.

У загальному випадку задача формулюється таким чином: знайти значення $x_1, x_2, \ldots, x_n$, що задовольняють систему з лінійних алгебраїчних рівнянь (СЛАР, system of linear algebraic equations):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n, \end{cases} \tag{11.1}$$

або в матричній формі $AX = B$, де

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \qquad \mathrm{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Методи розв'язання систем лінійних рівнянь можна розділити на прямі та ітераційні. До прямих, які дозволяють одержати точний розв'язок, відносяться методи визначників Крамера, Гауса, прогонки. Ітераційні методи, що ґрунтуються на одержанні і уточненні послідовних наближень до точного розв'язку, ефективні в тому випадку, коли є багато нульових коефіцієнтів або високий порядок системи (метод Гауса ефективний до порядку 10^4, ітераційні – до 10^6).

До найбільш популярних прямих методів відносять *метод Гауса* та його різновиди.

Цей метод є одним з найпоширеніших методів рішення СЛАР. У його основі лежить ідея послідовного виключення невідомих, що приводить вихідну систему до трикутного виду, у якому всі коефіцієнти нижче головної діагоналі дорівнюють нулю. Існують різні обчислювальні схеми, що реалізують цей метод. Класичний метод Гауса (метод виключення) ґрунтується на доведенні матриці коефіцієнтів системи (11.1) до трикутного вигляду:

$$\begin{vmatrix} * & * & * & \ldots & * & * \\ 0 & * & * & \ldots & * & * \\ 0 & 0 & * & \ldots & * & * \\ \vdots & \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & * & * \\ 0 & 0 & 0 & \ldots & 0 & * \end{vmatrix}$$

і складається з двох етапів: прямого ходу і зворотного підставлення. Етап прямого ходу закінчується, коли одне з рівнянь системи стає рівнянням з одним невідомим. Далі, здійснюючи зворотне підставлення, знаходять всі невідомі.

Алгоритм методу Гауса.

Ділимо перше рівняння на a_{11}, після чого воно отримує вигляд:

$$x_1 + \frac{a_{12}}{a_{11}}x_2 + \ldots + \frac{a_{1n}}{a_{11}}x_n = \frac{b_1}{a_{11}} \tag{11.2}$$

Помножимо послідовно (11.2) на a_{21}, a_{31}, …, a_{n1} та віднімемо відповідно з 2-го, … , n-го рівняння системи. Отримаємо СЛАР:

$$\begin{cases} x_1 + a_{12}^1 x_2 + \ldots + a_{1n}^1 x_n = b_1^1 \\ 0 + a_{22}^1 x_2 + \ldots + a_{2n}^1 x_n = b_2^1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0 + a_{n2}^1 x_2 + \ldots + a_{nn}^1 x_n = b_n^1 \end{cases}$$

Її елементи розраховані за формулами:

$$a_{ij}^1 = a_{ij} - a_{1j}^1 a_{i1} \qquad \text{для}$$

$$b_i^1 = b_i - b_1^1 a_{i1} \qquad i = \overline{2,n} \quad j = \overline{1,n}$$

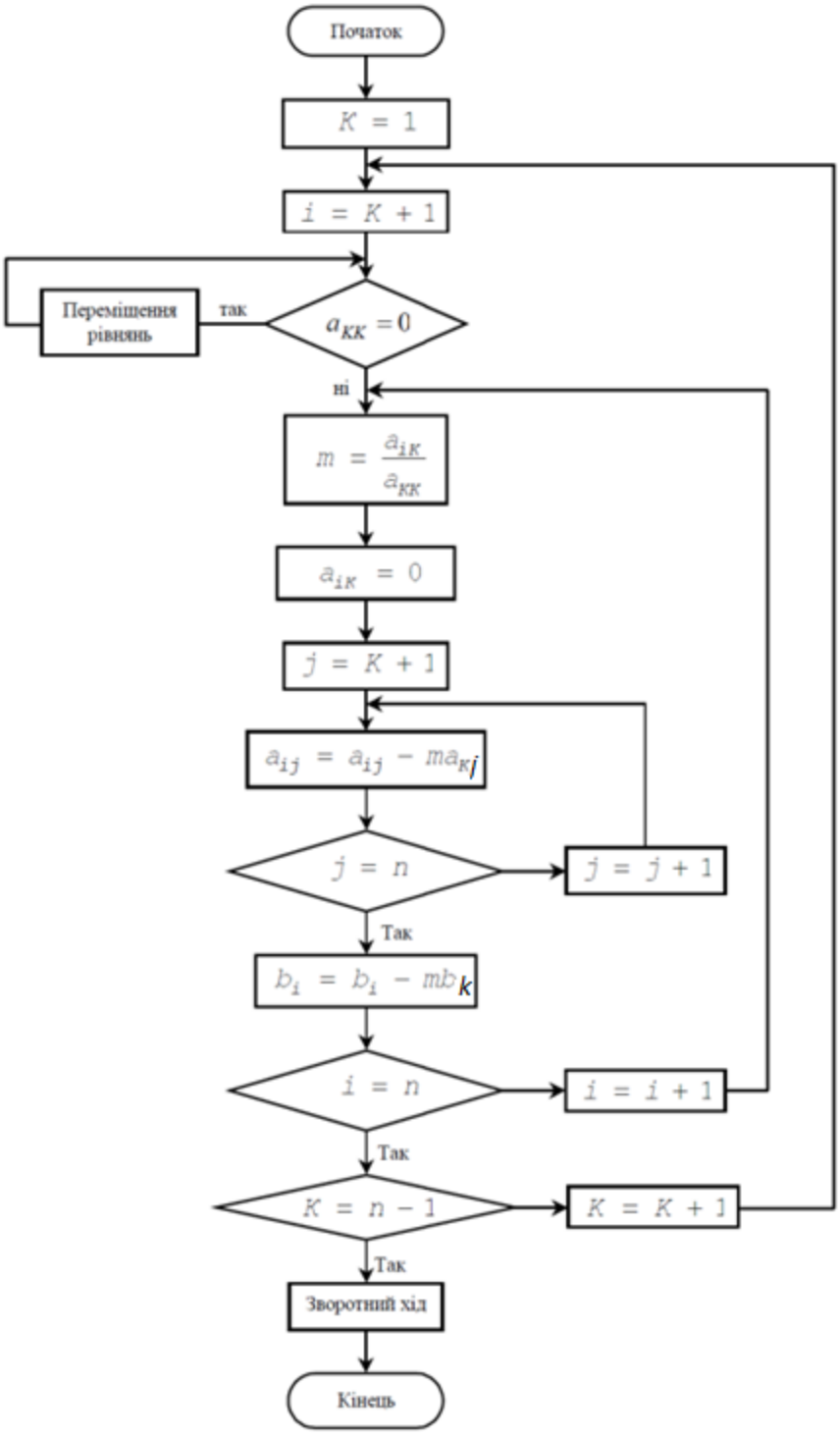


Рис. 11.2. Блок-схема алгоритму методу Гауса.

Прямий хід реалізуємо за формулами:

$$\left.\begin{aligned} a_{ij}^{*} &= a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj}, \quad k=\overline{1,n-1}; \quad j=\overline{k,n}; \\ b_{i}^{*} &= b_{i} - \frac{a_{ik}}{a_{kk}} b_{k}, \quad i=\overline{k+1,n} \end{aligned}\right\}$$

де i – номер рівняння, з якого виключається x_k;

k – номер невідомого, яке виключається з решти (n - k) рівнянь, а також позначає номер рівняння, за допомогою якого виключається x_k;

j – номер стовпця вихідної матриці;

a_{kk} – головний (ведучий) елемент матриці. Під час рахунку необхідно стежити, щоб $a_{kk} \neq 0$. В іншому випадку вдаються до перестановки рядків матриці.

Для реалізації наведених формул у програмі використовуємо *три вкладених цикли*:

зовнішній цикл по k від 1 до $n-1$, другий цикл по i від $k+1$ до n та внутрішній цикл по j від k до n.

З метою зменшення обчислювальної похибки використовують метод Гауса з вибором головного елементу. Під головним елементом будемо розуміти максимальний по модулю елемент матриці A, обраний по заданій множині рядків та стовпців. Спочатку знаходиться головний елемент матриці А і шляхом перестановки рівнянь та стовпців він розміщується на місці елемента a_{11} (перестановку стовпців потрібно запам'ятовувати для вірного співставлення змінних з отриманими значеннями).

У випадку, коли проблеми ділення на нуль не виникало і була отримана трикутна матриця A^n, система має єдиний розв'язок. Для його пошуку застосовується ***зворотній хід*** метода Гауса:

$$x_n = \frac{b'}{a'_{nn}};$$

$$x_k = \frac{1}{a'_{kk}}\left[b'_k - \sum_{i=k+1}^{n} a'_{ki}\, x_i \right], \quad k=\overline{n-1,1}.$$

Завдання до лабораторної роботи

Рівень А. Знайдіть значення вказаного матричного виразу.

11.1А.

$$A=\begin{pmatrix}3&-1&2\\-2&0&3\\1&2&4\end{pmatrix};\ B=\begin{pmatrix}0&2&3\\1&3&5\\-1&0&1\end{pmatrix};\ A^TB-B^TA.$$

11.2А.

$$A=\begin{pmatrix}-2&-3&1\\0&1&4\\2&-2&0\end{pmatrix};\ B=\begin{pmatrix}-2&6&-2\\7&0&3\\-1&3&9\end{pmatrix};\ AB-BA.$$

11.3А.

$$A=\begin{pmatrix}-1&1&0\\2&1&5\\-1&0&-4\end{pmatrix};\ B=\begin{pmatrix}2&7&1\\-5&2&0\\0&-2&8\end{pmatrix};\ (AB)^T-B^TA^T.$$

11.4А.

$$A=\begin{pmatrix}0&3&-2\\0&3&1\\3&4&-3\end{pmatrix};\ B=\begin{pmatrix}2&-3&1\\-5&3&2\\2&0&7\end{pmatrix};\ AB-BA.$$

11.5А.

$$A=\begin{pmatrix}-5&3&1\\0&2&-4\\-1&0&3\end{pmatrix};\ B=\begin{pmatrix}-6&0&4\\2&3&7\\2&-1&3\end{pmatrix};\ (A+B)(B-A)+A^2-B^2.$$

11.6А.

$$A=\begin{pmatrix}0&2&-5\\-2&3&1\\2&0&5\end{pmatrix};\ B=\begin{pmatrix}9&-2&0\\0&-3&2\\-4&1&7\end{pmatrix};\ (AB)^T-B^TA^T.$$

11.7А.

$$A=\begin{pmatrix}-7&0&-2\\0&2&-3\\-1&3&9\end{pmatrix};\ B=\begin{pmatrix}2&2&-1\\-3&2&0\\4&-1&7\end{pmatrix};\ A^TB-B^TA.$$

11.8А.

$$A=\begin{pmatrix}1&0&3\\7&-2&1\\3&2&-6\end{pmatrix};\ B=\begin{pmatrix}-2&5&-2\\1&2&3\\-3&0&-1\end{pmatrix};\ (A+B)(B-A)+A^2-B^2.$$

11.9А.

$$A=\begin{pmatrix}8&0&-2\\1&-3&7\\0&-2&-4\end{pmatrix};\ B=\begin{pmatrix}-3&1&-7\\1&3&-8\\0&1&-4\end{pmatrix};\ AB-BA.$$

11.10A.

$$A=\begin{pmatrix}2&3&-1\\4&1&-2\\5&0&2\end{pmatrix};\ B=\begin{pmatrix}0&-1&3\\2&3&-3\\-4&0&-2\end{pmatrix};\ A^TB-B^TA.$$

11.11A.

$$A=\begin{pmatrix}2&3&-2\\0&-4&3\\-1&2&3\end{pmatrix};\ B=\begin{pmatrix}1&4&-5\\3&1&-3\\2&0&-6\end{pmatrix};\ (AB)^T-B^TA^T.$$

11.12A.

$$A=\begin{pmatrix}4&0&1\\1&2&-3\\-7&1&5\end{pmatrix};\ B=\begin{pmatrix}2&1&-3\\0&3&-4\\2&-2&0\end{pmatrix};\ (A+B)(B-A)+A^2-B^2.$$

11.13A.

$$A=\begin{pmatrix}0&1&-2\\3&5&7\\-1&0&-4\end{pmatrix};\ B=\begin{pmatrix}2&-3&5\\2&7&0\\8&-2&1\end{pmatrix};\ AB-BA.$$

11.14A.

$$A=\begin{pmatrix}3&5&0\\0&7&2\\-1&-2&3\end{pmatrix};\ B=\begin{pmatrix}4&-3&-2\\2&3&5\\0&3&-1\end{pmatrix};\ A^TB-B^TA.$$

11.15A.

$$A=\begin{pmatrix}1&-6&4\\3&-2&5\\2&1&4\end{pmatrix};\ B=\begin{pmatrix}0&3&2\\1&5&0\\-1&2&1\end{pmatrix};\ (AB)^T-B^TA^T.$$

11.16A.

$$A=\begin{pmatrix}-5&0&-2\\2&3&0\\-1&-3&3\end{pmatrix};\ B=\begin{pmatrix}4&1&3\\0&-1&2\\9&-2&0\end{pmatrix};\ (A+B)(B-A)+A^2-B^2.$$

11.17A.

$$A=\begin{pmatrix}0&2&-2\\4&3&0\\-1&-2&4\end{pmatrix};\ B=\begin{pmatrix}-1&3&7\\0&1&2\\0&-2&3\end{pmatrix};\ AB-BA.$$

11.18A.

$$A=\begin{pmatrix}4&2&-1\\-7&2&0\\5&-2&3\end{pmatrix};\ B=\begin{pmatrix}2&-3&4\\0&1&-3\\-1&3&0\end{pmatrix};\ A^TB-B^TA.$$

11.19A.

$$A=\begin{pmatrix}4&-6&-1\\3&2&-1\\-4&3&0\end{pmatrix};\ B=\begin{pmatrix}0&1&-3\\3&-2&5\\6&0&1\end{pmatrix};\ (AB)^T-B^TA^T.$$

Рівень В. Розв'язати систему лінійних рівнянь за методом Гауса..

11.1В.

$$\begin{cases} x_1 + 2x_2 + 4x_3 = 8, \\ -2x_1 + 3x_2 = 13, \\ x_1 + 4x_2 - 2x_3 = 8. \end{cases}$$

11.2В.

$$\begin{cases} 3x_1 + 2x_2 - x_3 = -4, \\ 4x_1 - 3x_2 - 4x_3 = -14, \\ -2x_1 + x_2 + 3x_3 = 5. \end{cases}$$

11.3В.

$$\begin{cases} 2x_1 + 2x_2 + 3x_3 = 10, \\ -3x_1 + 4x_2 + 5x_3 = -3, \\ x_1 - 2x_2 - 2x_3 = 1. \end{cases}$$

11.4В.

$$\begin{cases} 2x_1 - 3x_2 + 3x_3 = -1, \\ 7x_1 + 3x_2 - 2x_3 = 7, \\ 4x_1 + x_2 + 3x_3 = -7. \end{cases}$$

11.5В.

$$\begin{cases} 2x_1 + 3x_2 + x_3 = -5, \\ 3x_1 - 4x_2 - 5x_3 = 5 \\ 2x_1 + 2x_2 + 3x_3 = 2. \end{cases}$$

11.6В.

$$\begin{cases} 3x_1 + x_2 - x_3 = -1, \\ 2x_1 - 3x_2 - 3x_3 = 1, \\ -3x_1 + 2x_2 + 2x_3 = -4. \end{cases}$$

11.7В.

$$\begin{cases} 2x_1 - 2x_2 + 3x_3 = 1, \\ 3x_1 - 5x_2 + 4x_3 = 5, \\ 4x_1 + 2x_2 + 15x_3 = -1. \end{cases}$$

11.8В.

$$\begin{cases} 2x_1 + 3x_2 + 2x_3 = -8, \\ 3x_1 + 3x_2 + 2x_3 = -5, \\ 4x_1 + x_2 + 2x_3 = 2. \end{cases}$$

11.9В.

$$\begin{cases} 2x_1 + 3x_2 + 4x_3 = 6, \\ x_1 + 2x_2 + x_3 = 7, \\ 5x_1 + x_2 + x_3 = 7. \end{cases}$$

11.10В.

$$\begin{cases} -2x_1 - 5x_2 + x_3 = 12, \\ 2x_1 + 4x_2 + 3x_3 = 6, \\ 3x_1 + x_2 + x_3 = 5. \end{cases}$$

11.11В.

$$\begin{cases} x_1 + 2x_2 - x_3 = 11, \\ 2x_1 - x_2 - 3x_3 = 4, \\ 7x_1 - 2x_2 + x_3 = -3. \end{cases}$$

11.12В.

$$\begin{cases} 5x_1 + x_2 + x_3 = 7, \\ 3x_1 - 3x_2 - x_3 = 5, \\ 2x_1 + 6x_2 + 3x_3 = 2. \end{cases}$$

3. Передача масивів у функції

Іноді може бути потрібним передати масив у функцію у вигляді параметру. В С++ не можливо передати повний блок даних у пам'яті по значенню як параметр, але можливо передати його адресу. На практиці це майже те саме, і це навіть більш швидка і ефективна операція. Для того, щоб функція могла прийняти масив як параметр, при оголошенні функції ми повинні для даного параметру вказати тип елементів масиву, ідентифікатор і порожні квадратні дужки [].

Наприклад:

```
int sum_array (int arg[])
```

приймає в якості параметра масив цілих чисел з ідентифікатором arg.

Код, який передаватиме в цю функцію масив буде виглядати так:

```
int myarray [5] = { 1, 2, 4, 8, 16 };

sum_array (myarray);
```

При використовуванні масиву як параметра функції передається покажчик на його перший елемент, тобто масив завжди передається за адресою. При цьому інформація про кількість елементів втрачається і слід передавати його розмірність через окремий параметр. Якщо розмірність масиву є константою, проблем не виникає, оскільки можна зазначити її при оголошенні формального параметра і як верхню межу циклів при опрацюванні масиву всередин функції.

Тобто, оскільки нам треба знати довжину масиву, щоб використати у циклі, ми передаємо її за допомогою другого параметру int length.

Приклад 11.3.

```
#include <iostream>

using namespace std;

int sum_array (int arg[], int length)
{
  int result = 0;
  for (int i=0; i<length; i++)
  {
         result = result + arg[i];
  }
  return result;
}
int main()
{
    int myarray [5] = { 1, 2, 4, 8, 16 };

  cout << "Suma = " << sum_array (myarray,5)<< endl;

    return 0;
}
```

```
Suma = 31
```

При передаванні до функції *багатовимірних масивів* усі розмірності, якщо вони невідомі на етапі компіляції, мають передаватися як параметри.

Якщо обидві розмірності є відомі й є константами, то заголовок функції , яка обчислює суму елементів двовимірного масиву, матиме вигляд

```
int sum(int a[4][6]);
```

Розглянемо кілька варіантів передавання матриці цілих чисел `A[3][4]` у функцію `print()`, яка організовує виведення цієї матриці на екран у консолі.

1) Якщо розміри обох індексів є відомі, то проблем немає:

```
void print(int A[3][4])
{ for(int i=0; i<3; i++)
  { for(int j=0; j<4; j++) cout << ' ' << A[i][j];
    cout << '\n';
} }
```

Матриця і тут передається як покажчик, а розмірності наведено просто для повноти опису. Виклик такої функції може бути таким:

```
int x[3][4];
print(x);
```

2) Перша розмірність для обчислення адреси елемента є неважлива, тому її можна передавати як параметр:

```
void print(int A[][4], int m)
{ for(int i=0; i<m; i++)
  { for(int j=0; j<4; j++) cout << ' ' << A[i][j];
    cout << '\n';
} }
```

Виклик цієї функції:

```
int x[3][4];
print(x,3);
```

3) Найскладніший випадок – коли треба передавати обидві розмірності.

Наведений нижче код функції є поширеною помилкою:

```
void print(int A[][], int m, int n)   // Помилка!
{ for(int i=0; i<m; i++)
  { for(int j=0; j<n; j++) cout << ' ' << A[i][j];
    cout << '\n';
} }
```

По-перше, опис параметра A[][] є неприпустимий, оскільки для обчислення адреси елемента двовимірного масиву слід знати другу розмірність. У такому разі найбільш поширеним і грамотним є застосування динамічних масивів. Наведемо правильний код такої функції:

```
void print(int** A, int m, int n)
{ for(int i=0; i<m; i++)
  { for(int j=0; j<n; j++)
    cout << ' ' << A[i][j];
    cout << '\n';
  }
}
```

Виклик такої функції матиме вигляд:

```
int **x=new int*[3];
for(i=0; i<3; i++) x[i]=new int[4];
print(x,3,4);
```

Сенс простий: кількість зірочок змінної покажчика залежить від мірності масиву (одномірні - одна, двовимірні - дві і т. д.).

Завдання до лабораторної роботи

Рівень С. Реалізувати метод Гауса у вигляді окремої функції.