Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

СТРУКТУРИ ДАНИХ

Методичні рекомендації
до практичних занять з курсу «Програмування» для студентів спеціальностей 113 «Прикладна математика», 142 «Енергетичне машинобудування», 143 «Атомна енергетика», 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Електронне видання

Харків – 2023

УДК: 519.6: 519.7: 621.039: 681.3
С 87

Рецензенти:

Є. С. Меняйлов – кандидат технічних наук, доцент кафедри теоретичної та прикладної математики Харківського національного університету імені В. Н. Каразіна;

О. М. Цимбал – доктор технічних наук, професор кафедри комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки Харківського національного університету радіоелектроніки.

Затверджено до розміщення в мережі Інтернет рішенням Науково-методичної ради Харківського національного університету імені В. Н. Каразіна (протокол № 9 від 16.06.2023 р.)

Структури даних : методичні рекомендації до практичних занять з курсу «Програмування» для студентів спеціальностей 113 «Прикладна математика», 142 «Енергетичне машинобудування», 143 «Атомна енергетика», 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» [Електронне видання] / уклад. В. І. Коробов, Ю. В. Ромашов, К. В. Стєпанова. – Харків : ХНУ імені В. Н. Каразіна, 2023. – (PDF 32 с.)

С 87

Методичні рекомендації підготовлені для вивчення теми «Структури даних» відповідно до змісту навчальної дисципліни «Програмування». Обговорюються базові принципи щодо використання масивів заданої та змінної довжини, а також типових прийомів програмування щодо обробки масивів та розв'язування систем лінійних алгебраїчних рівнянь.

Методичні рекомендації призначені для здобувачів вищої освіти різних курсів та рівнів за спеціальностями 113 «Прикладна математика», 142 «Енергетичне машинобудування», 143 «Атомна енергетика», а також 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка».

УДК 519.6: 519.7: 621.039: 681.3

Електронне навчальне видання комбінованого використання
Можна використовувати в локальному та мережному режимі

Коробов Валерій Іванович
Ромашов Юрій Володимирович
Стєпанова Катерина Вадимівна

СТРУКТУРИ ДАНИХ

Методичні рекомендації
до практичних занять з курсу «Програмування» для студентів спеціальностей 113 «Прикладна математика», 142 «Енергетичне машинобудування», 143 «Атомна енергетика», 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Коректор *О. В. Анцибора*
Комп'ютерне верстання *К. В. Стєпанова*

Підписано до розміщення 17.06.2023. Гарнітура Times New Roman.
Ум. друк. арк. 1,90. Обсяг 0,353 Мб. Зам. 116/23.

Харківський національний університет імені В. Н. Каразіна,
61022, м. Харків, майдан Свободи, 4.
Свідоцтво суб'єкта видавничої справи ДК № 3367 від 13.01.2009.
Видавництво ХНУ імені В. Н. Каразіна

ЗМІСТ

ВСТУП

Комп'ютеризація (digitalization) сьогодні є одним із ключових засобів забезпечення сталого розвитку суспільства та економіки, тому до неї приділяється багато уваги в розвинених країнах, в країнах ЄС [1] тощо. Комп'ютеризація передбачає розробку й впровадження відповідних програмних засобів, у тому числі для наукової та інжинірингової діяльності, що забезпечує розв'язування багатьох сучасних глобальних проблем [1]. Отже, вивчення програмування є важливим для навчання майже за кожною спеціальністю. Ці методичні вказівки призначені для здобувачів вищої освіти різних курсів та рівнів за спеціальностями 113 Прикладна математика, 142 Енергетичне машинобудування, а також 143 Атомна енергетика й 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка, які хоча й відносяться до різних галузей, але мають багато спільних інтересів щодо математичного забезпечення програмних засобів різного призначення, у тому числі для переходу до кліматично нейтральної енергетики, що є однією із важливих глобальних проблем сучасності та одним із пріоритетів діяльності ЄС [1] тощо.

Широке впровадження комп'ютерів та програмних продуктів для розв'язування складних задач суспільства, економіки, наукових та інженерних досліджень, промисловості природно призвело до суттєвого ускладнення розроблюваних програм. Одним із характерних джерел такого ускладнення програм є необхідність використовувати дуже велику кількість змінних різних типів для опису досліджуваних складних об'єктів (ситуацій). Використання структур даних при розробці програм дозволяє спростити маніпуляції із великими обсягами даних.

Метою цих методичних рекомендацій до практичних занять з курсу «Програмування» є засвоєння теоретичного матеріалу щодо базових понять про структури даних, а також їхнє використання при розробці програм для наукових та інженерних розрахунків.

Реалізація зазначеної мети буде здійснюватися шляхом поступового створення від простіших до більш складних програм щодо виконання наукових та інженерних розрахунків, в яких принципово необхідно використовувати структури даних у вигляді масивів. Буде використано мову програмування FORTRAN 77, яка є зручною для первинного навчання програмуванню, оскільки на основі цієї мови програмування створено велику кількість різноманітних сучасних мов програмування, та завдяки широкому використанню сьогодні удосконалених версій FORTRAN щодо розробки програм для наукових та інженерних розрахунків [2, 3]. Для виконання завдань рекомендується використовувати доступний для вільного користування компілятор проекту Open Watcom, який можна без обмежень завантажувати із мережі Internet [4, 5].

1 МАСИВИ

Особливий тип даних, який складається із великих обсягів даних одного типу або різних типів та надає можливості доступу до цих даних, – це структури даних; структури даних складаються, звичайно, із попередньо введених простих типів даних. Завдяки використанню структур даних, інформацію про складний об'єкт можна представляти однією змінною – структурою даних – та мати доступ до усіх змінних, що визначають цей об'єкт.

Простішою структурою даних є масив, який насправді широко використовується в програмах різного призначення, особливо для наукових та інженерних розрахунків, і передбачений у різних виглядах в різних мовах програмування.

1.1 Теоретичні відомості про масиви

Розглянемо далі базові поняття про масиви та їх використання в програмах. Цілком зрозуміло, що структуризація даних в програмах повністю узгоджується із концепціями структурного програмування [6].

1.1.1 Масив являє собою множину елементів одного типу, які об'єднані одним іменем таким чином, що кожному елементу масиву ставиться у відповідність ціле число або декілька упорядкованих цілих чисел. Доступ до елементу масиву здійснюється шляхом визначення імені масиву та індексу (індексів) його елементу. Використання масивів надає можливості доступу до великої кількості даних через одну змінну, що буває зручним, а іноді просто необхідним, в деяких ситуаціях.

Формальне визначення масиву в програмуванні здійснюється як відображення упорядкованої множини цілих чисел на множину значень заданого типу [6, 7]:

$$\left(i_1 \in I_1\right)\times\left(i_2 \in I_2\right)\times\ldots\times\left(i_N \in I_N\right) \to a_{i_1 i_2 \ldots i_N} \in D_a\,, \qquad (1.1)$$

де i_1, i_2,..., i_N – цілочислові значення та I_1, I_2,..., I_N – області визначення цих значень; $a_{i_1 i_2 \ldots i_N}$ – елементи масиву та D_a – область їхнього визначення.

Цілочислові значення i_1, i_2,..., i_N, що визначають елемент масиву, називають індексами. Кожен індекс визначає окремий вимір масиву; кількість елементів масиву в деякому вимірі називають довжиною масиву в цьому вимірі. З точки зору визначення (1.1) вектори, що використовують в лінійній алгебрі, є одновимірними (N = 1) масивами дійсних чисел, довжина яких відповідає розмірності цих векторів. При цьому матриці, що використовують в лінійній алгебрі, є двовимірними (N = 2) масивами дійсних чисел, довжини яких відповідають розмірам цих матриць. Зрозуміло,

але слід підкреслити, що масив, як і будь-яка інша структура даних, складається із попередньо передбачених в мові програмування створених типів даних – цілочислових значень індексів та типу, що визначає область D_a визначення елементів масиву.

1.1.2 Для завдання масиву, зрозуміло, необхідно визначити тип його елементів, а також кількість його вимірів та довжину в кожному із цих вимірів, що робиться відповідним чином в різних мовах програмування. В мові програмування FORTRAN масиви визначаються наступним чином [2, 3]:

```
type name(i1min:i1max,i2min:i2max,...,inmin:inmax)
```

де `type` – це визначення типу елементів масиву; `name` – ім'я масиву; `i1min` та `i1max` – мінімальне та максимальне значення індексу в першому вимірі масиву; `i2min` та `i2max` – мінімальне та максимальне значення індексу в другому вимірі масиву і таким же чином до `inmin` та `inmax` – мінімальне та максимальне значення індексу у n-му вимірі масиву.

Кількість вимірів масиву визначається кількістю пар мінімальних і максимальних значень індексів в описі масиву, тобто є на одиницю більшою, ніж кількість ком в дужках опису масиву. Якщо мінімальне значення індексу в деякому напрямку дорівнює одиниці, то його можна не вказувати, а вказувати лише максимальне значення індексу без двокрапки. Так, одновимірні масиви `X` та `B` дійсних чисел завдовжки 20 елементів, а також двовимірний масив `A` завдовжки 20 елементів у кожному із вимірів можуть бути визначені, наприклад, так:

```
CCC   ARRAYS DEFINITIONS
      REAL X(0:19)
      REAL B(20)
      REAL A(20,20)
```

Масиви з елементами однакового типу можна визначати в одному рядка, що економить розмір тексту програми:

```
CCC   ARRAYS DEFINITIONS
      REAL X(0:19), B(20), A(20,20)
```

Доступ до елементів масивів у програмах здійснюється шляхом вказування імені масиву та індексів необхідного елементу в круглих дужках через коми; посилаючись таким чином на елемент масиву, його можна використовувати як звичайну змінну, наприклад, у арифметичних виразах:

```
...
      REAL X(0:19), B(20), A(20,20)
...
      X(0)=25.0
      A(1,1)=3.5
```

Підкреслимо, що елементи масивів можуть мати не тільки дійсний тип, а й будь-який стандартний (цілочисловий та логічний) тип, що є у FORTRAN.

1.1.3 Масиви можна передавати в якості параметрів до підпрограм, що широко використовується при розробці програм різного призначення та, особливо, бібліотек підпрограм. Якщо до підпрограми передається масив, то в цій підпрограмі має обов'язково міститись опис цього масиву, який має бути цілком відповідним до опису в програмній компоненті, що викликала підпрограму [2, 3].

1.2 Завдання щодо програм із використанням масивів

Реалізацію розглянутих загальних рекомендацій щодо використання масивів у програмах далі покажемо на прикладі розв'язування системи лінійних алгебраїчних рівнянь.

1.2.1 Лінійне алгебраїчне рівняння має вигляд:

$$ax = b, \tag{1.2}$$

де a, b – задані числові параметри рівняння та x – шукана невідома.

Запис одного рівняння (1.2) є дуже зрозумілим, але спроба розглянути систему вже двох лінійних рівнянь приводить до доцільності використання індексів для визначення числових значень параметрів рівнянь:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1, \\ a_{21}x_1 + a_{22}x_2 &= b_2, \end{aligned} \tag{1.3}$$

де a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2 – задані числові параметри системи рівнянь та x_1, x_2 – шукані невідомі.

Систему рівнянь (1.3) можна записати і без індексних позначень із використанням необхідної кількості літер латинського алфавіту. Для систем з насправді великою кількістю n лінійних алгебраїчних рівнянь використання індексів є принциповим:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots a_{1n}x_n = b_1, \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots a_{2n}x_n = b_2, \\ &\ldots \\ &a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nj}x_j + \ldots a_{nn}x_n = b_n, \end{aligned} \tag{1.4}$$

де позначення числових параметрів системи та шуканих невідомих аналогічно простішій системі (1.3).

1.2.2 Інформацію про систему лінійних алгебраїчних рівнянь (1.4) зручно представити у вигляді таблиці A та стовпця b, а інформацію про шукані невідомі – відповідно у вигляді стовпця *x*:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}. \tag{1.5}$$

Таблицю чисел вигляду A в математиці називають матрицею, а стовпці вигляду b та *x* – векторами. Доступ до кожного елементу векторів b та *x* можна здійснити за допомогою одного індексу, що вказує номер відповідного елементу, тобто сукупність елементів векторів можна визначити за допомогою цілочислового індексу, наприклад, індексу *i*, який має змінюватися від одиниці до *n*, що можна представити як відображення множини цілих чисел $i \in \{1,2,\dots,n\}$ на множину значень елементів векторів b та *x:*

$$i \to b_i, \quad i \to x_i, \qquad i = 1,2,\dots n\,. \tag{1.6}$$

Доступ до кожного елементу матриці A можна здійснювати за допомогою упорядкованої пари цілочислових індексів, що визначають номер рядка та номер стовпця, наприклад індексів *i* та *j*, які мають змінюватися від одиниці до *n*, що можна представити у вигляді відображення множини упорядкованої пари $i \times j$ цілих чисел $i \in \{1,2,\dots,n\}$ та $j \in \{1,2,\dots,n\}$ на множину значень елементів матриці A:

$$i \times j \to a_{ij}, \quad i, j = 1,2,\dots n\,. \tag{1.7}$$

Вимога до упорядкованості пари $i \times j$ індексів необхідна, щоб відрізняти елементи a_{ij} та a_{ji}. Слід підкреслити, що визначення векторів (1.6) та матриці (1.7) є окремими випадками загального визначення (1.1) масиву.

1.3 Приклади програм з масивами

Розглянемо далі приклади програм, в яких для зберігання даних використовуються масиви.

1.3.1 Як приклад розглянемо програму (ліст. 1.1) розв'язування системи двох лінійних алгебраїчних рівнянь (1.3). Ця програма має розпочинатися із введення матриці та вектора правої частини системи лінійних алгебраїчних рівнянь, що можна легко здійснити за допомогою відповідних циклів. У програмі (ліст. 1.1) передбачається послідовне введення

коефіцієнтів матриці та вектора правої частини для рівнянь системи. Розв'язування здійснюється методом Крамера.

Лістинг 1.1 – Розв'язування системи двох лінійних алгебраїчних рівнянь

```
      PROGRAM LAES2
      REAL A(2,2),B(2),X(2),DETA,DETA1,DETA2
      INTEGER I,J
      DO I=1,2
      DO J=1,2
      PRINT '(''A('',I1,'','',I1,'')=''$)',I,J
      READ *,A(I,J)
      END DO
      PRINT '(''B('',I1,'')=''$)',I
      READ *,B(I)
      END DO
      DETA=A(1,1)*A(2,2)-A(2,1)*A(1,2)
      DETA1=B(1)*A(2,2)-B(2)*A(1,2)
      DETA2=A(1,1)*B(2)-A(2,1)*B(1)
      X(1)=DETA1/DETA
      X(2)=DETA2/DETA
CCCCCC RESULTS OUTPUTTING
      DO I=1,2
      DO J=1,2
      PRINT '(F10.4,''X'',I1,\)',A(I,J),J
      IF (J.LT.2) PRINT'(''+''\)'
      END DO
      PRINT '(''='',F10.4)',B(I)
      END DO
      DO I=1,2
      PRINT '(''X('',I1,'')='',F10.4)',I,X(I)
      END DO
      PAUSE 'PRESS ENTER TO EXIT PROGRAM'
      END
```

1.3.2 Наведена вище на ліст. 3.1 програма може бути представлена у вигляді 4-х підпрограм, а саме – введення матриці та вектора системи рівнянь, розв'язку системи рівнянь, друку системи рівнянь та друку вектора, як показано, наприклад, на ліст. 1.2. Цілком природно, що використання таких підпрограм потребуватиме використання масивів в якості параметрів цих підпрограм. За цих умов у таких підпрограмах відповідним чином передбачене визначення типів усіх масивів, які передбачені в якості параметрів підпрограм. Зрозуміло, що запропоновані підпрограми (ліст. 1.2) можуть використовувати лише одновимірні та двовимірні масиви завдовжки 2 в кожному вимірі.

Лістинг 1.2 – Підпрограми для маніпуляцій із системами двох лінійних алгебраїчних рівнянь (продовження – на наступній стор.)

```
PROGRAM LAES2
REAL A(2,2),B(2),X(2)
CALL INPUTLAES(A,B)
CALL PRINTLAES(A,B)
CALL SOLVELAES(A,B,X)
CALL PRINTVECTOR(X)
PAUSE 'PRESS ENTER TO EXIT PROGRAM'
END
SUBROUTINE INPUTLAES(A,B)
REAL A(2,2),B(2)
INTEGER I,J
DO I=1,2
DO J=1,2
PRINT '(''A('',I1,'','',I1,'')=''$)',I,J
READ *,A(I,J)
END DO
PRINT '(''B('',I1,'')=''$)',I
READ *,B(I)
END DO
END
 SUBROUTINE PRINTLAES(A,B)
 REAL A(2,2),B(2)
 INTEGER I,J
 DO I=1,2
 DO J=1,2
 PRINT '(F10.4,''X'',I1,$)',A(I,J),J
 IF (J.LT.2) PRINT'(''+''$)'
 END DO
 PRINT '(''='',F10.4)',B(I)
 END DO
 END
SUBROUTINE SOLVELAES(A,B,X)
REAL A(2,2),B(2),X(2),D,D1,D2
D=A(1,1)*A(2,2)-A(2,1)*A(1,2)
D1=B(1)*A(2,2)-B(2)*A(1,2)
D2=A(1,1)*B(2)-A(2,1)*B(1)
X(1)=D1/D
X(2)=D2/D
END
```

```
SUBROUTINE PRINTVECTOR(X)
REAL X(2)
INTEGER I
DO I=1,2
PRINT '(''X('',I1,'')='',F10.4)',I,X(I)
END DO
END
```

1.4 Контрольні запитання та завдання

Наявність знань, що підтверджують засвоєння матеріалу заняття, можна перевірити спроможністю надати відповіді та виконати наступні питання та завдання:

1. Що таке структури даних і для чого вони потрібні в програмах?
2. Що таке масиви?
3. Яким чином визначають та використовують масиви в програмах на мови програмування FORTRAN?
4. Поясніть доцільність використання масивів при написанні програм щодо розв'язування систем лінійних алгебраїчних рівнянь.
5. Доповніть програму, що наведена вище на ліст. 1.2, додатковою підпрограмою-функцією, яка здійснюватиме обчислення визначника матриці розміром 2×2, використати цю підпрограму щодо розв'язування системи двох лінійних алгебраїчних рівнянь.

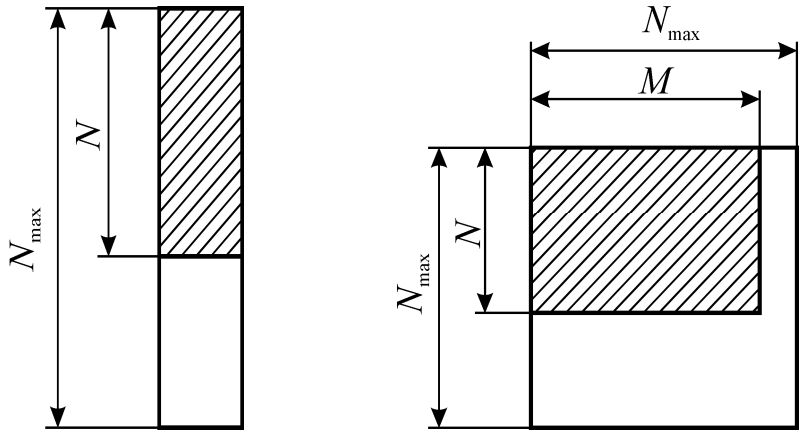
2 МАСИВИ ЗМІННОЇ ДОВЖИНИ ТА ВИКОРИСТАННЯ СТРУКТУР ДАНИХ

Дуже часто довжини масивів визначаються в програмах у залежності від поточного стану певних змінних та, навіть можуть змінюватися протягом роботи програми.

2.1 Теоретичні відомості про масиви змінної довжини

Розглянемо далі деякі базові питання щодо прийомів програмування масивів змінної довжини, яка визначається та, навіть, може змінюватися протягом роботи програми.

2.1.1 Найпростішим, але нераціональним з точки зору використання пам'яті, способом роботи із масивами змінної довжини є створення масиву, довжина $N_{\max}$ якого заздалегідь перевищує довжини масивів, що будуть використовуватися у програмі, та визначення розмірів масиву в кожному окремому випадку. Реалізацію такого простішого підходу на прикла-



а) б)

а) одновимірний масив (вектор); *б) двовимірний масив (матриця)*

Рисунок 2.1 – Використання частини масиву в якості масиву

ді одновимірного масиву змінної довжини N та двовимірного масиву змінної довжини $N \times M$ можна наочно уявляти, як це показано на рис. 2.1. Звісно, що при цьому пам'ять комп'ютера використовується неефективно, оскільки для розміщення масивів в деяких випадках може виділятися суттєво більше пам'яті, ніж це потрібно, і ця виділена пам'ять не може бути використана.

Відсоток M_{NU} невикористаної пам'яті при застосуванні масивів завідомо більшої довжини для забезпечення роботи із масивами змінної довжини для одновимірного та двовимірного масивів визначається відповідно так:

$$M_{NU} = \left(1 - \frac{N}{N_{\max}}\right) \cdot 100\%, \tag{2.1}$$

$$M_{NU} = \left(1 - \frac{N \cdot M}{N_{\max} \cdot N_{\max}}\right) \cdot 100\%. \tag{2.2}$$

Звісно, що використання масивів завідомо більшої довжини в якості масивів змінної довжини в більшості випадків веде до нераціонального використання пам'яті комп'ютера. В той же час, простота програмної реалізації робить такий підхід дуже привабливим, і його використання є цілком виправданим для випадків, коли M_{NU} є достатньо низьким, що для одновимірного масиву буде виконуватися, коли $N \sim N_{\max}$, а для одновимірного – коли $N \sim N_{\max}$ та $M \sim N_{\max}$. В деяких програмах існує можливість обрати $N_{\max}$ таким чином, щоб M_{NU} була досить малою, і завдяки цьому використання масивів завідомо більшої довжини в якості масивів змінної довжини в таких програмах є цілком виправданим.

Величину $N_{\max}$ зручно ввести із використанням інструкції

```
PARAMETER (NMAX=100)
```

Ця інструкція має бути розташованою на початку тексту програмної одиниці перед інструкціями опису типів даних та дозволяє шляхом однократної зміни розширювати або зменшувати максимальну границю масивів. Зрозуміло, що величину, яка введена за допомогою інструкції

PARAMETER, заборонено змінювати в програмі; в цій інструкції також можна вводити інші значення простих типів, не обов'язково границі довжин масивів. Значення, що введені за допомогою цієї інструкції PARAMETER, визначені виключно у відповідній програмній компоненті.

2.1.2 Зрозуміло, що не завжди можна запропонувати одне значення $N_{\max}$ таким чином, щоб M_{NU} була достатньо малою для усіх масивів змінної довжини, що будуть використовуватися у програмі. В цих випадках зручним є використання масивів невизначеної довжини. Такий масив визначається лише ім'ям відповідної йому змінної та кількістю вимірів за допомогою символів ":" через кому у круглих дужках одразу ж після імені змінної [2, 3]. Далі у програмі за допомогою інструкції ALLOCATE виділяють пам'ять для розміщення цього масиву відповідно до необхідних довжин у кожному із вимірювань та працюють із таким масивом як із звичайним масивом [2, 3]. Після завершення роботи із таким масивом слід за допомогою інструкції DEALLOCATE звільнити пам'ять, що була необхідна для розташування масиву, і далі можна знов виділяти пам'ять на цей масив відповідно до нових потрібних довжин в кожному із вимірювань [2, 3]. Для одновимірного та двовимірного масиву невизначеної довжини визначення, виділення пам'яті, робота та звільнення пам'яті може бути здійснено, наприклад, наступним чином:

```
...
      REAL B(:),A(:,:)
...
      ALLOCATE(B(5))
      ALLOCATE(A(2,7))
...
      B(1)=1.5
      A(2,5)=B(1)
...
      DEALLOCATE (A,B)
...
```

При використанні масивів невизначеної довжини в підпрограмах в якості вхідних параметрів підпрограм слід передавати довжини масиву у всіх його вимірах, але в самій підпрограмі визначати таки масиви як масиви із заданими довжинами, що відповідають відповідним вхідним параметрам підпрограми. У порівнянні із технологією програмування масивів змінної довжини на основі використання завідомо більшої довжини, використання масивів невизначеної довжини є більш ефективним з точки зору використання пам'яті комп'ютера, але потребує від програміста пам'ятати про необхідність виділення та звільнення пам'яті у необхідних місцях програми. Слід зазначити, що простота програмування масивів

є однією із вагомих переваг мови програмування FORTRAN перед іншими мовами програмування, але ця перевага є принциповою тільки при розробці програм матричних обчислень, що потребується переважно для наукових та інженерних розрахунків.

2.1.3 При роботі із масивами необхідно мати інформацію щодо довжин масивів в усіх вимірах, для чого, зазвичай, передбачають відповідну кількість змінних. Якщо в програмі використовується велика кількість масивів різної довжини, то іноді можна заплутатися в іменах змінних, що визначають довжини цих масивів. Шляхом спеціального підходу щодо вибору імен змінних можна уникнути плутанини щодо відповідності між іменами змінних та іменами масивів, довжину яких ці змінні визначають. В той же час, відповідно до концепції структурного програмування [6] для визначення масивів бажано мати таку структуру даних, в якій визначені довжини масиву та його елементи, що дозволило б повністю представляти масиви у вигляді однієї змінної. Можливості щодо використання такого підходу надають, наприклад, розширення стандартної мови програмування FORTRAN 77, які передбачені в компіляторі WATCOM, у вигляді спеціальних нестандартних інструкцій `STRUCTURE` та `RECORD`, які дозволяють створювати та використовувати в програмах складні структури, які містять в собі дані різного типу. Підкреслимо, що використання таких складних структур даних є принциповою ідеєю об'єктно-орієнтованого програмування, в якому передбачається об'єднання між собою не тільки даних, але також передбачається й об'єднання даних із підпрограмами (методами), які призначені для роботи з цими даними.

Нестандартна інструкція `STRUCTURE` створює новий тип даних, який містить змінні різного типу, кожна із яких називається полем структури та буде локально визначеною виключно усередині структури, в якій вона визначена; інструкція `END STRUCTURE` завершує визначення нового типу даних. Інструкція `RECORD` створює змінну нестандартного типу, визначеного за допомогою інструкції `STRUCTURE`. Доступ до полів структури даних здійснюється вказуванням імені змінної структури даних, символу "%" або ".", а також імені поля структури даних. Типові використання інструкцій роботи із структурами даних показані нижче:

```
...
      STRUCTURE /name/
...
      INTEGER I,J
...
      END STRUCTURE
...
      RECORD /name/ varname
```

```
...
      varname.I=5
...
      varname%J=varname%I
...
```

При використанні структури даних для визначення двовимірного масиву до полів цієї структури даних доцільно ввести довжини масиву в кожному із його вимірів, а також масив із необхідною кількістю вимірів для зберігання елементів. Це дозволить в одній змінний, що має відповідний структурний тип, зберігати усі дані, які повністю визначають масив.

2.2 Приклади використання масивів змінної довжини

Використання масивів змінної довжини є притаманним програмним компонентам, які, наприклад, забезпечують введення та виведення матриць при виконанні наукових та інженерних розрахунків.

2.2.1 Приклад програми, в якій здійснюється введення та друк довжин та елементів двовимірного масиву змінної довжини, який представляє деяку матрицю, є таким:

Лістинг 2.1 – Приклад використання масивів завідомо більшої довжини в якості масивів змінної довжини

```
      PROGRAM ARRAY3
      PARAMETER (NMAX=50)
      REAL M(NMAX,NMAX)
      INTEGER MN,MM
      CALL INPUTMATRIX(M,MN,MM,NMAX)
      CALL PRINTMATRIX(M,MN,MM,NMAX)
      PAUSE 'PRESS ENTER TO EXIT PROGRAM'
      END
      SUBROUTINE INPUTMATRIX(A,N,M,SIZE)
      INTEGER N,M,SIZE,I,J
      REAL A(SIZE,SIZE)
      PRINT '(''INPUT COUNT ROWS OF MATRIX: '',$)'
      READ *,N
      PRINT '(''INPUT COUNT COLS OF MATRIX: '',$)'
      READ *,M
      DO I=1,N
      PRINT '(''INPUT ROW '',I2,'':'')',I
      DO J=1,M
      PRINT
     1'(''INPUT ELEM('',I2,'','',I2,'')='',$)',I,J
```

```
 READ *,A(I,J)
 END DO
 END DO
 END
 SUBROUTINE PRINTMATRIX(A,N,M,SIZE)
 INTEGER N,M,SIZE,I,J
 REAL A(SIZE,SIZE)
 PRINT '(''COUNT ROWS OF MATRIX, ROWS='',I2)',N
 PRINT '(''COUNT COLS OF MATRIX, COLS='',I2)',M
 DO I=1,N
 PRINT '(''ROW = '',I2)',I
 DO J=1,M
 PRINT '(''ELEM('',I2,'','',I2,'')='',F10.4)'
1,I,J,A(I,J)
 END DO
 END DO
 END
```

2.2.2 Використання масивів невизначеної довжини, окрім раціонального використання пам’яті, дозволяє також створювати більш універсальні програмні одиниці:

Лістинг 2.2 – Приклад використання масивів невизначеної довжини

```
 PROGRAM ARRAY4
 REAL M(:,:)
 INTEGER MN,MM
 CALL IMPUTMATRIXSIZES(MN,MM)
 ALLOCATE(M(MN,MM))
 CALL INPUTMATRIXELEM(M,MN,MM)
 CALL PRINTMATRIX(M,MN,MM)
 DEALLOCATE(M)
 PAUSE 'PRESS ENTER TO EXIT PROGRAM'
 END

 SUBROUTINE IMPUTMATRIXSIZES(N,M)
 INTEGER N,M
 PRINT '(''INPUT COUNT ROWS (N): '',\)'
 READ *,N
 PRINT '(''INPUT COUNT COLS (M): '',\)'
 READ *,M
 END
 SUBROUTINE INPUTMATRIXELEM(A,N,M)
 INTEGER N,M,I,J
 REAL A(N,M)
 DO I=1,N
```

```
 PRINT '(''INPUT ROW '',I2)',I
 DO J=1,M
 PRINT '(''ELEM ('',I2,'','',I2,'')='',\)',I,J
 READ *,A(I,J)
 END DO
 END DO
 END
 SUBROUTINE PRINTMATRIX(A,N,M)
 INTEGER N,M,SIZE,I,J
 REAL A(N,M)
 PRINT '(''COUNT COLS OF MATRIX, N='',I2)',N
 PRINT '(''COUNT COLS OF MATRIX, M='',I2)',M
 DO I=1,N
 PRINT '(''ROW = '',I2)',I
 DO J=1,M
 PRINT '(''ELEM('',I2,'','',I2,'')='',F10.4)'
1,I,J,A(I,J)
 END DO
 END DO
 END
```

2.2.3 Приклад можливого використання спеціально створених структур даних для зберігання двовимірного масиву – матриці є наступним:

Лістинг 2.3 – Використання структур даних для визначення матриці

```
 PROGRAM ARRAY5
 PARAMETER (NMAX=50)
 STRUCTURE /MATRIX/
 INTEGER ROWS,COLS
 REAL ELEMENT(NMAX,NMAX)
 END STRUCTURE
 RECORD /MATRIX/ M
 CALL INPUTMATRIX(M%ELEMENT,M%ROWS,M%COLS,NMAX)
 CALL PRINTMATRIX(M.ELEMENT,M.ROWS,M.COLS,NMAX)
 PAUSE 'PRESS ENTER TO EXIT PROGRAM'
 END
```

Підпрограми, що використовуються в програмі, яка наведена на ліст. 2.3, є точно такими, як у програмі, що наведена вище на ліст. 2.1. Зрозуміло, що для забезпечення можливостей зміни розмірів масиву в програмі, що наведена на ліст. 2.3, використовуються масиви із завідомо більшою довжиною. На жаль, масиви невизначеної довжини не можуть бути використані в якості полів структур інструкції STRUCTURE, що є суттєвим обмеженням можливостей FORTRAN 77, навіть із розширеннями.

2.3 Контрольні запитання та завдання

Наявність знань, що підтверджують засвоєння матеріалу заняття, можна перевірити спроможністю надати відповіді та виконати наступні питання та завдання:

1. На прикладі розв'язування систем лінійних алгебраїчних рівнянь поясніть доцільність використання масивів змінної довжини в програмах.
2. В чому полягає ідея використання масивів завідомо більшої довжини в якості масивів змінної довжини та в чому полягають недоліки такого підходу?
3. Яким чином в мові програмування FORTRAN передбачено використання масивів невизначеної довжини?
4. Що таке підпрограма-процедура, як вона оформлюється в мові програмування FORTRAN та як її використовувати в програмах?
5. Доповнити програму, що показана вище на ліст. 2.2, підпрограмами, що здійснюватимуть введення та друк вектора.

3 ТИПОВІ ПРИЙОМИ ПРОГРАМУВАННЯ ЩОДО ОБРОБКИ МАСИВІВ

Розв'язування багатьох задач, у тому числі виконання наукових та інженерних розрахунків, зводиться до відповідної спеціальної обробки масивів.

3.1 Теоретичні відомості про типові прийоми обробки масивів

В мовах програмування, та в FORTRAN у тому числі, передбачені різноманітні засоби, що спрощують обробку масивів в програмах.

3.1.1 Слід зазначити, що для визначення довжин використовуваних масивів у програмних одиницях у багатьох випадках досить зручно використовувати інструкцію `PARAMETER`, яка передбачена в мові програмування FORTRAN та дозволяє визначати значення параметрів, які, на відміну від змінних, не можуть бути зміненими у програмі. Наприклад, таку інструкцію досить зручно використовувати в програмах для визначення кількості рядків та стовпців матриці наступним чином:

```
PARAMETER (N=5, M=10)
```

Зручність використання інструкції `PARAMETER` для визначення довжин масивів обумовлена тим, що значення визначених в ній параметрів можна використовувати в описах масивів. Завдяки цьому забезпечується можливе спрощення модифікацій розроблених програм у подальшому.

3.1.2 В багатьох випадках програма містить дані, які є відомими заздалегідь. Такі дані можна визначати шляхом присвоювання необхідних значень відповідним змінним, але у випадку великої кількості таких даних в деяких випадках зручним є використання передбаченої в мові програмування FORTRAN спеціальної інструкції `DATA`, що дозволяє у зручній формі здійснити присвоєння значень та має вигляд:

```
DATA vlist/clist/
```

де `vlist` – список змінних, значення яких мають бути присвоєні; `clist` – список констант, що послідовно відповідають значенням змінних із попереднього списку `vlist`.

Сенс від використання інструкції `DATA` полягає в тому, щоб відділити у програмах присвоєння (ініціалізацію) початкових значень змінних від подальших їхніх змін протягом роботи програм. Зручності від використання цієї інструкції, яка вважається застарілою, насправді не є наочною взагалі, але ж в деяких випадках таке дійсно є доцільним.

3.2 Завдання щодо типових прийомів обробки масивів

Матриці широко використовуються для формулювання математичних задач. Числові характеристики матриці, такі як максимальний та мінімальний елементи матриці, її заданого рядка або стовпця, множення рядка або стовпця на задане число, досить широко використовуються в програмах різного призначення, для виконання наукових та інженерних розрахунків тощо.

Деякі числові характеристики матриці, такі як визначник або ранг, не змінюються при елементарних перетворюваннях, таких як множення рядка матриці на число, додавання до рядка матриці іншого рядка цієї ж матриці, та такі самі перетворення для стовпчиків матриці. Означені властивості елементарних перетворень матриць надають можливостей для їхнього перетворення до спеціального вигляду, наприклад із нульовими елементами, що розташовані під діагональними елементами:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & \dots & a_{1m} \\ 0 & a_{22} & a_{23} & a_{24} & a_{25} & \dots & a_{2m} \\ 0 & 0 & a_{33} & a_{34} & a_{35} & \dots & a_{3m} \\ 0 & 0 & 0 & a_{44} & a_{45} & \dots & a_{4m} \\ 0 & 0 & 0 & 0 & a_{55} & \dots & a_{5m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & a_{nm} \end{pmatrix}. \qquad (3.1)$$

Матриці спеціального вигляду, такі як, наприклад, (3.1), мають зрозумілі властивості, що дозволяють дуже просто визначати їхні характеристики, такі як, наприклад, визначник та ранг. Перетворення матриці довільного вигляду до вигляду (3.1) дозволяє розв'язувати багато прикладних задач.

3.3 Приклади програмування типових прийомів обробки масивів

Розглянемо далі типові задачі щодо обробки двовимірного масиву розміром 3×4 наступного вигляду:

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 2 & 3 \\ 4 & 2 & 2 & 3 & 4 \\ 2 & 1 & 1 & 1 & 1 \end{pmatrix}. \qquad (3.2)$$

3.3.1 На ліст. 3.1 наведено текст програми, в якій передбачається ініціалізація даних двовимірного масиву, що представляє матрицю (3.2), за допомогою інструкції DATA, та запропоновано підпрограму виведення двовимірного масиву на екран у зручному вигляді таблиці рядків та стовпчиків.

Лістинг 3.1 – Ініціалізація значень та виведення масиву на екран

```
 PROGRAM RANK
 INTEGER N,M
 PARAMETER (N=4,M=5)
 REAL A(N,M)
 DATA A/2*2.0,4.0,2.0,2*1.0,2.0,1.0,2*1.0,2.0
1,1.0,1.0,2.0,3.0,1.0,1.0,3.0,4.0,1.0/
 CALL PRINTMATRIX(N,M,A)
 PAUSE 'PRESS ENTER TO EXIT'
 END
 SUBROUTINE PRINTMATRIX(N,M,A)
 INTEGER N,M
 REAL A(N,M)
 PRINT '(A)','---------- MATRIX DATA :'
 PRINT 100,'COUNT ROWS=',N
 PRINT 100,'COUNT COLS=',M
 PRINT '(A)','..ELEMENTS='
 DO I=1,N
 DO J=1,M
 IF (J.LT.M) THEN
 PRINT '(1X,E12.5,\)',A(I,J)
```

```
      ELSE
      PRINT '(1X,E12.5)',A(I,J)
      END IF
      END DO
      END DO
      PRINT '(A)','-------------------------'
100   FORMAT(A,I2)
      END
```

3.3.2 Для визначення максимального елементу рядка слід у циклі порівняти між собою значення елементів цього рядка, як це зроблено у відповідній підпрограмі на ліст. 3.2. Ця підпрограма (ліст. 3.2) має отримати дані щодо розмірів масиву, елементів масиву та номера рядка, в якому слід знайти максимальний елемент, щоб в результаті визначити максимальне значення серед елементів заданого рядка та відповідний цьому максимальному значенню номер елементу рядка. В прикладі використання цієї підпрограми (ліст. 3.2) у відповідному циклі здійснюється визначення та виведення на екран монітора результатів щодо максимального елементу усіх рядків двовимірного масиву.

3.3.3 Розв'язування багатьох задач вимагає перестановки рядків двовимірного масиву. Для цього слід передбачити окрему допоміжну змінну, щоб зберегти дані від знищення, як у підпрограмі на ліст. 3.3. Ця підпрограма має отримати дані щодо розмірів масиву, елементів масиву, а також номерів рядків масиву, які слід переставити, і в результаті цього елементи заданих рядків будуть переставлені безпосередньо в заданому масиві. Використання цієї підпрограми показано на ліст. 3.3, де здійснюється виведення вихідного масиву, перестановка 2-го та 4-го рядків масиву та виведення масиву з переставленими рядками.

Лістинг 3.2 – Пошук максимальних елементів рядків двовимірного масиву

```
      PROGRAM RANK
      INTEGER N,M,NM
      PARAMETER (N=4,M=5)
      REAL A(N,M),AM
      DATA A/2*2.0,4.0,2.0,2*1.0,2.0,1.0,2*1.0,2.0
     1,1.0,1.0,2.0,3.0,1.0,1.0,3.0,4.0,1.0/
      CALL PRINTMATRIX(N,M,A)
      DO I=1,N
      CALL MAXROW(N,M,A,I,AM,NM)
      PRINT
     1'(''ROW='',I1,2X,''MAX='',E12.5,2X,''N='',I1)'
     2,I,AM,NM
```

```
      END DO
      PAUSE 'PRESS ENTER TO EXIT'
      END
CCCCCC FINDING THE MAXIMUM ELEMENT OF THE ROW
      SUBROUTINE MAXROW(N,M,A,ROW,VALUE,NUMBER)
      INTEGER N,M,ROW,NUMBER
      REAL A(N,M),VALUE
      VALUE=A(ROW,1)
      NUMBER=1
      DO I=1,M
      IF (VALUE.LT.A(ROW,I)) THEN
      VALUE=A(ROW,I)
      NUMBER=I
      END IF
      END DO
      END
```

3.3.4 Можлива реалізація перетворення матриці довільного вигляду до стандартного вигляду (3.1), яке дозволяє розв'язувати багато задач, як показана на ліст. 3.4. Перетворення, що необхідні для обчислення рангу матриці, є ідентичними до перетворень, що необхідні для розв'язування системи лінійних алгебраїчних рівнянь, будуть більш докладно розглянуті у подальшому. Зараз рекомендується підготувати програму відповідно до ліст. 3.4 та подивитися результат її виконання, щоб побачити її можливості щодо обчислення рангу матриці.

Лістинг 3.3 – Перестановка рядків двовимірного масиву

```
      PROGRAM RANK
      INTEGER N,M,NM
      PARAMETER (N=4,M=5)
      REAL A(N,M),AM
      DATA A/2*2.0,4.0,2.0,2*1.0,2.0,1.0,2*1.0,2.0
     1,1.0,1.0,2.0,3.0,1.0,1.0,3.0,4.0,1.0/
      CALL PRINTMATRIX(N,M,A)
      CALL PERMUTROWS(N,M,A,4,2)
      CALL PRINTMATRIX(N,M,A)
      PAUSE 'PRESS ENTER TO EXIT'
      END
      SUBROUTINE PERMUTROWS(N,M,A,R1,R2)
      INTEGER N,M,R1,R2
      REAL A(N,M),EL
      DO I=1,M
      EL=A(R1,I)
```

```
A(R1,I)=A(R2,I)
A(R2,I)=EL
END DO
END
```

3.4 Контрольні запитання та завдання

Наявність знань, що підтверджують засвоєння матеріалу заняття, можна перевірити спроможністю надати відповіді та виконати наступні питання та завдання:

1. Для чого призначена передбачена в мові програмування FORTRAN інструкція `PARAMETER` та як її використовувати в програмах?
2. Для чого призначена передбачена в мові програмування FORTRAN інструкція `DATA` та як її використовувати в програмах?
3. Які числові характеристики матриць та які перетворення матриць часто використовуються при розв'язуванні прикладних задач?
4. Наведену на ліст. 3.2 програму доповнити підпрограмами визначення мінімального елементу рядка, а також максимального та мінімального елементів стовпця двовимірного масиву.
5. Наведену на ліст. 3.3 програму доповнити підпрограмою перестановки стовпців двовимірного масиву.
6. Наведений на ліст. 3.4 програмний код, який здійснює перетворення, необхідні для визначення рангу матриці, оформити у вигляді підпрограми.

Лістинг 3.4 – Перетворення, що необхідні для визначення рангу матриці

```
PROGRAM RANK
PARAMETER (N=4,M=5)
REAL A(N,M),AM
INTEGER NM
DATA A/2*2.0,4.0,2.0,2*1.0,2.0,1.0,2*1.0,2.0
1,1.0,1.0,2.0,3.0,1.0,1.0,3.0,4.0,1.0/
CALL PRINTMATRIX(N,M,A)
DO I=1,N
AM=ABS(A(I,I))
NM=I
DO J=I,N
IF (AM.LT.ABS(A(J,I))) THEN
AM=ABS(A(J,I))
NM=J
END IF
END DO
```

```
IF (NM.NE.I) THEN
DO J=I,M
AM=A(I,J)
A(I,J)=A(NM,J)
A(NM,J)=AM
END DO
END IF
IF (A(I,I).NE.0.0) THEN
DO J=I+1,N
AM=A(J,I)/A(I,I)
DO K=I,M
A(J,K)=A(J,K)-A(I,K)*AM
END DO
END DO
END IF
END DO
CALL PRINTMATRIX(N,M,A)
PAUSE 'PRESS ENTER TO EXIT'
END
```

4 РОЗВ'ЯЗУВАННЯ СИСТЕМИ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ

Багато теоретичних та прикладних задач із різних галузей знань може бути зведено до розв'язування систем лінійних алгебраїчних рівнянь, тому питання щодо програмування такого розв'язування є досить важливими та заслуговують окремої уваги.

4.1 Теоретичні відомості про метод Гауса

Простішим для розуміння методом розв'язування системи лінійних алгебраїчних рівнянь є метод Гауса, який зводиться до перетворень матриць за спеціальними алгоритмами.

4.1.1 В загальному вигляді систему лінійних алгебраїчних рівнянь (1.3) зазвичай представляють за допомогою матриці та векторів (1.5). Для програмування методу Гауса систему лінійних алгебраїчних рівнянь зручно представляти за допомогою розширеної матриці:

$$\mathbf{A_b} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{pmatrix}, \qquad (4.1)$$

яка має розмір $n\times\left(n+1\right)$, що визначається кількістю m рівнянь у розв'язуваній системі та складається із матриці та вектора (1.5), які визначають цю систему.

Ідея методу Гауса щодо розв'язування системи лінійних алгебраїчних рівнянь полягає у послідовному виключенні невідомих із використанням рівнянь системи: з другого та усіх подальших рівнянь за допомогою першого рівняння виключають першу невідому; далі із третього рівняння та усіх подальших рівнянь за допомогою другого рівняння виключають другу невідому і так далі до останнього рівняння, в якому залишатиметься одна невідома. Останнє рівняння з однією невідомою легко розв'язується, а всі інші невідомі послідовно визначаються починаючи із передостанньої до першої.

4.1.2 Цілком зрозуміло, що метод Гауса можна уявляти як так званий «прямий хід» – тотожні перетворення розширеної матриці (4.1) до спеціального вигляду:

$$\mathbf{A}'_{\mathbf{b}}=\begin{pmatrix}1 & a'_{12} & \dots & a'_{1n} & b'_1\\ 0 & 1 & \dots & a'_{2n} & b'_2\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \dots & 1 & b'_n\end{pmatrix}. \tag{4.2}$$

Тоді визначення розв'язку відповідної системи рівнянь, що тотожно перетворена до вигляду (4.2), можна здійснити наступним чином:

$$x_i=b'_i-\sum_{j=i+1}^{n}a'_{ij}x_j,\quad i=n,n-1,n-2,\dots,1. \tag{4.3}$$

Таким чином, розв'язування системи лінійних алгебраїчних рівнянь методом Гауса здійснюється в два етапи:

1) програмування «прямого ходу» – тотожних перетворень розширеної матриці (4.1), до спеціального вигляду (4.2);

2) програмування «зворотного ходу» – визначення розв'язку за формулами (4.3).

Зрозуміло, що програмування методу Гауса також здійснюється шляхом програмування означених двох етапів.

4.1.3 Щоб тотожно перетворити перший рядок розширеної матриці (4.1) до спеціального вигляду (4.2), необхідно поділити кожний елемент першого рядка на елемент a_{11}, що математично записується так:

$$\overline{a}_{1j}=a_{1j},\ j=1,2,...,n,\ \overline{b}_1=b_1,\quad a_{1j}=\frac{\overline{a}_{1j}}{\overline{a}_{11}},\ j=1,2,...,n,\ b_1=\frac{\overline{b}_1}{\overline{a}_{11}}. \tag{4.4}$$

В результаті виконання перетворення (4.4) матриця (4.1) набуде вигляду:

$$\begin{pmatrix} 1 & \alpha_{12} & \dots & \alpha_{1n} & \beta_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{pmatrix}, \tag{4.5}$$

де $\alpha_{1j} = a_{1j} / a_{11}$, $j = 2,3,...,n$, $\beta_1 = b_1 / a_{11}$.

Зрозуміло, що алгоритм (4.4) не спрацює у випадку, коли $a_{11} = 0$, тому перед тим, як здійснювати цей алгоритм, слід визначити максимальний за модулем елемент першого стовпця матриці та переставити між собою перший рядок та рядок, що містить в собі максимальний за модулем елемент. Після перетворення вихідної матриці (4.1) до вигляду (4.5) далі шляхом множення першого рядка на елемент a_{j1}, $j = 2,3,...,n$ та віднімання результату такого множення із відповідного рядка $j = 2,3,...,n$ можемо привести матрицю (4.5) до наступного вигляду:

$$\begin{pmatrix} 1 & \alpha_{12} & \dots & \alpha_{1n} & \beta_1 \\ 0 & \alpha_{22} & \dots & \alpha_{2n} & \beta_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \alpha_{n2} & \dots & \alpha_{nn} & \beta_n \end{pmatrix}, \tag{4.6}$$

де $\alpha_{ij} = a_{ij} - \alpha_{1j} a_{i1}$, $\beta_i = b_i - \beta_1 a_{i1}$, $i, j = 2,3,...,n$.

В системі рівнянь з матрицею (4.6) в другому та в усіх подальших рівняннях виключено першу невідому. Аналогічним чином виконаємо перетворення матриці

$$\begin{pmatrix} \alpha_{22} & \alpha_{23} & \dots & \alpha_{2n} & \beta_2 \\ \alpha_{32} & \alpha_{33} & \dots & \alpha_{3n} & \beta_{32} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{n2} & \alpha_{n3} & \dots & \alpha_{nn} & \beta_n \end{pmatrix}, \tag{4.7}$$

яка має розмір $(n-1) \times n$. Далі будемо розглядати матрицю розміром $(n-2) \times (n-1)$ і т.д. впритул до матриці розміром 1×2 . Звісно, що в програмі, яка реалізує метод Гауса, немає необхідності вводити масиви для усіх цих «проміжних» матриць, оскільки кожна із них є частиною вихідної матриці та розташована у відповідній частині масиву, в якому зберігається уся матриця системи. Для цього потрібно реалізувати цикл по кількості рівнянь, та кожний наступний крок цього циклу реалізовувати для коефіцієнтів, індекси яких перевищують індекс цього циклу.

4.2 Завдання щодо розв'язування лінійних алгебраїчних рівнянь

Розроблювані далі програми будемо використовувати та перевіряти на прикладі розв'язування системи лінійних алгебраїчних рівнянь із матрицею Гілберта [8]

$$a_{ij} = \frac{1}{i+j-1}, \quad i, j = 1, 2, ..., n. \tag{4.8}$$

Вибір матриці (4.8) узгоджений із досвідом щодо розробки та тестування програм для розв'язування системи лінійних алгебраїчних рівнянь [8].

В якості вектора правої частини вибираємо наступний:

$$b_i = -\sum_{j=1}^{n} j \cdot a_{ij}, \quad i = 1, 2, ..., n. \tag{4.9}$$

Вектору (3.25) відповідає розв'язок

$$x_i = i, \quad i = 1, 2, ..., n. \tag{4.10}$$

Отже, маємо дані (4.10) для тестування розроблюваних програм.

4.3 Приклади програмування етапів методу Гауса

Програмування методу Гауса при вивченні програмування є цілком виправданим через можливості щодо придбання простіших навичок та початкового досвіду реалізації алгоритмів обробки масивів, які є типовими для програм різного призначення.

4.3.1 Програмний код, що реалізує перетворення (4.4) першого рядка розширеної матриці системи лінійних алгебраїчних рівнянь, може мати наступний вигляд:

Лістинг 4.1 – Перетворення першого рядка розширеної матриці

```
...
REAL A(N,N),B(N),AII
INTEGER J
...
AII=A(1,1)
DO J=1,N
A(1,J)=A(1,J)/AII
END DO
B(1)=B(1)/AII
...
```

Тотожні перетворення матриці (4.1) щодо пошуку максимального елементу стовпця та перестановки відповідних рядків на прикладі першого рядка показані на ліст. 4.2. Після такої перестановки рядків можемо перетворити перший рядок розширеної матриці до вигляду як в (4.5) за допомогою програмного коду, показаного на ліст. 4.1.

Лістинг 4.2 – Пошук максимального елементу та перестановка рядків

```
      REAL A(N,N),B(N),AII,ABSA
      INTEGER J,K
      ...
      AII=ABS(A(1,1))
      K=1
      DO J=2,N
      ABSA=ABS(A(J,1))
      IF (ABSA.GT.AII) THEN
      AII=ABSA
      K=J
      END IF
      END DO
      IF (K.NE.1) THEN
      DO J=1,N
      AII=A(1,1)
      A(1,1)=A(K,1)
      A(K,1)=AII
      END DO
      AII=B(1)
      B(1)=B(K)
      B(K)=AII
```

4.3.2 З урахуванням досвіду розробки програм, наведених вище на ліст. 4.1 та ліст. 4.2, можемо запропонувати програму, що реалізує прямий хід методу Гауса у наступному вигляді:

Лістинг 4.3 – Прямий хід методу Гауса (продовження – на наст. стор.)

```
      LOGICAL FUNCTION GAUSS1(A,B,N)
      INTEGER N,I,J,K
      REAL A(N,N),B(N),AII,ABSA
      DO I=1,N
CCC   MAX ELEMENT FINDING FOR COLUMN NUMBER I
      AII=ABS(A(I,I))
      K=I
      DO J=I+1,N
      ABSA=ABS(A(J,I))
      IF (ABSA.GT.AII) THEN
```

```
      AII=ABSA
      K=J
      END IF
      END DO
      IF (AII.EQ.0) THEN
      GAUSS1=.FALSE.
      GOTO 100
      END IF
CCC   REARRANGEMENT ROW I AND ROW K IF K NOT EQUAL I
      IF (K.NE.I) THEN
      DO J=I,N
      AII=A(I,J)
      A(I,J)=A(K,J)
      A(K,J)=AII
      END DO
      AII=B(I)
      B(I)=B(K)
      B(K)=AII
      END IF
CCC   DIVIDING ROW I
      AII=A(I,I)
      DO J=I,N
      A(I,J)=A(I,J)/AII
      END DO
      B(I)=B(I)/AII
CCC   NULING ROWS I+1,I+2,..., N IN COLUMN I
      DO K=I+1,N
      AII=A(K,I)
      DO J=I,N
      A(K,J)=A(K,J)-A(I,J)*AII
      END DO
      B(K)=B(K)-B(I)*AII
      END DO
      END DO
      GAUSS1=.TRUE.
100   END
```

Вибір підпрограми-функції для реалізації прямого ходу методу Гауса потрібен для повідомлення про випадок, коли детермінант матриці системи дорівнює нулю, який призводить до помилки ділення на нуль.

4.3.3 Програма, що реалізує зворотний хід (4.3) методу Гауса, може мати вигляд:

Лістинг 4.4 – Зворотний хід методу Гауса

```
      SUBROUTINE GAUSS2(A,B,N)
```

```
INTEGER N,I,J
REAL A(N,N),B(N)
DO I=N,1,-1
DO J=N,I+1,-1
B(I)=B(I)-A(I,J)*B(J)
END DO
END DO
END
```

4.3.4 Програма, в якій здійснюється розв'язування системи лінійних алгебраїчних рівнянь із матрицею (4.8) для заданого вектора (4.9), може бути реалізована наступним чином:

Лістинг 4.5 – Використання підпрограм розв'язування систем лінійних алгебраїчних рівнянь (продовження – на наст. стор.)

```
PROGRAM GAUSS
INTEGER N,I,J
REAL A(:,:), B(:)
LOGICAL GAUSS1
N=3
ALLOCATE(A(N,N),B(N))
DO I=1,N
B(I)=0
DO J=1,N
A(I,J)=1.0/(I+J-1)
B(I)=B(I)+J*A(I,J)
END DO
END DO
CALL PRINTSYSTEM(A,B,N)
IF (GAUSS1(A,B,N)) THEN
CALL PRINTSYSTEM(A,B,N)
CALL GAUSS2(A,B,N)
CALL PRINTSYSTEM(A,B,N)
ELSE
PRINT *,'DETERMINANT IS ZERO'
END IF
DEALLOCATE(A,B)
PAUSE 'PRINT ENTER TO EXIT'
END

SUBROUTINE PRINTSYSTEM(A,B,N)
INTEGER N,I,J
REAL A(N,N),B(N)
PRINT*,'SYSTEM:'
```

```
      DO I=1,N
      DO J=1,N
      PRINT '(F10.4,$)',A(I,J)
      END DO
      PRINT '(F10.4)',B(I)
      END DO
      END

      LOGICAL FUNCTION GAUSS1(A,B,N)
      ...
100   END

      SUBROUTINE GAUSS2(A,B,N)
      ...
      END
```

В цій програмі зворотний хід методу Гауса реалізується тільки за умов, колі при виконанні прямого ходу не виявилося, що детермінант матриці системи дорівнює нулю. В програмі передбачений друк вихідної матриці та матриці після прямого та зворотного ходу методу Гауса, що дозволяє спостерігати за роботою програми.

Відомий розв'язок (4.10) дозволяє контролювати коректність роботи розроблених підпрограм, які реалізують метод Гауса щодо розв'язування систем лінійних алгебраїчних рівнянь. Завдяки цьому можна побачити, що розроблені підпрограми насправді надають можливість мати лише наближений розв'язок систем лінійних алгебраїчних рівнянь, але похибка такого наближеного розв'язку є досить малою, що дозволяє використовувати його в наукових та інженерних розрахунках. Також можемо побачити, що при збільшенні кількості розв'язуваних лінійних досить помітно збільшується похибка їхнього розв'язування за допомогою запропонованих підпрограм. Таке помітне збільшення похибки обумовлене властивостями матриці (4.8), тому саме таку матрицю досить часто використовують при тестуваннях програм, для розв'язування систем лінійних алгебраїчних рівнянь. У більшості прикладних задач маємо системи лінійних алгебраїчних рівнянь із такими матрицями, що не призводять до великих похибок наближених розрахунків, що отримані за допомогою програм, що реалізують метод Гауса.

4.4 Контрольні запитання та завдання

Наявність знань, що підтверджують засвоєння матеріалу заняття, можна перевірити спроможністю надати відповіді та виконати наступні питання та завдання:

1. В якому вигляді найбільш зручно уявляти систему лінійних алгебраїчних рівнянь?

2. В чому полягає ідея методу Гауса щодо розв’язування системи лінійних алгебраїчних рівнянь?

3. Як можна представити метод Гауса за допомогою перетворень розширеної матриці системи лінійних алгебраїчних рівнянь?

4. Чому при реалізації методу Гауса слід здійснювати пошук максимального елементу стовпця та здійснювати перестановку рядків матриці?

5. Зі скількох рівнянь складається система лінійних алгебраїчних рівнянь, що розв’язується в програмі, наведеній на ліст. 4.5?

6. Модифікувати текст підпрограми, наведений вище на ліст. 4.5, таким чином, щоб вона розв’язувала систему, яка містить п’ять лінійних алгебраїчних рівнянь, та оцінити похибку одержуваного розв’язку.

7. Модифікувати текст підпрограми, наведений вище на ліст. 4.5, таким чином, щоб вона розв’язувала систему, яка містить вісім лінійних алгебраїчних рівнянь, та оцінити похибку одержуваного розв’язку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ursula Gertrud von der Leyen. A Union that strives for more. My agenda for Europe: Political Guidelines for the Next European Commission 2019-2024. Available at:
https://commission.europa.eu/document/download/063d44e9-04ed-4033-acf9-639ecb187e87_en?filename=political-guidelines-next-commission_en.pdf [Accessed: 15.04.2023].

2. Chapman S.J. Fortran for Scientists and Engineers / S.J. Chapman. – New York: McGrawHill Education, 2018. – 1024 p.

3. Markus A. Modern Fortran in Practice / A. Markus. – New York: Cambridge University Pres, 2012. – 253 p.

4. http://openwatcom.org

5. https://github.com/open-watcom/open-watcom-1.9

6. Dahl O.-J. Structured programming / O.-J. Dahl, E.W. Dijkstra, C.A.R. Hoare. – London-New York: Academic Press, 1973. – 228 p.

7. Baase S. Computer algorithms. Introduction to Design and Analysis / S. Baase, A. Van Gelder. – 3rd ed. – New York: Addison-Wesley Longman, 2000. – 688 p.

8. Wilkinson J.H. Handbook for automatic computation. Linear algebra / J.H. Wilkinson, C. Reinsch. – Berlin-Heidelberg-New York: Springer-Verlag, 1971. – 439 p.