І. Семенова, М. Лавренюк

Завдання з програмування на фортрані

Київ 2012

УДК 519.682

Семенова І., Лавренюк М.

Завдання з програмування на фортрані: Навчальний посібник. – Київ: КНУ ім. Тараса Шевченка, 2012. – 84 с.

Викладено ключові питання мови програмування Фортран. Розглянуто важливі з точки зору реалізації при розв'язанні задач механіки твердого деформівного тіла методи та алгоритми обчислювальної математики та їх реалізацію на мові Фортран. Всі розділи супроводжуються як прикладами розв'язання задач так і завдань для самостійного опрацювання. Для студентів природничих факультетів університетів, спеціалістів-математиків і механіків.

Рецензенти: Лимарченко О. С.— д-р. техн. наук,
професор Київського національного університету
імені Тараса Шевченка;
Бєлов Ю. А. — д-р. фіз.-мат. наук,
професор Київського національного університету
імені Тараса Шевченка

Рекомендовано до друку Вченою радою
механіко-математичного факультету
Київського національного університету імені Тараса Шевченка
13.02.2012, протокол № 6

ЗМІСТ

Вступ

Традиційний підхід до вирішення завдань на комп'ютері є наступним: використовуючи бібліотечні процедури, скласти алгоритм, продумати введення і виведення, написати програму, скомпілювати, зібрати, налагодити, виконати і одержати результати.

Програмування задач на Фортрані в середовищі WinDOws. Назва мови Fortran походить від Formula Translator - перекладач формул. Не одне покоління програмістів виросло на Фортрані, і якщо комусь здається, що Фортран з минулого, то його новітні можливості, наприклад, для суперкомп'ютерів - якраз з майбутнього. Фортран критикували за «примітивність», але саме простота допомагає йому жити і розвиватися, зберігаючи спадкоємство і проходячи стандартизацію:
- 1954, Фортран - перша мова програмування, Джон Бекуса, IBM;
- Ф66 перший стандарт мови - універсальність і спадкоємність;
- Ф77 узвичаїв структурне програмування;
- Сучасний Фортран Ф90, Ф95 став віхою у розвитку мови:
- Масиви - опис, функції редукції, конструктори, секції, конформність масивів, виразів, розгалужень, циклів;
- Покажчики, рекурсивні процедури;
- Модульне програмування, механізми передачі даних;
- Ф03, Ф08 - об'єктно-орієнтоване програмування.
ФХХ - це Фортран по роках, жива класична мова програмування, стабільна, що розвивається, націлена в майбутнє, тільки в Фортрані:
- Історія налічує 56 років, чого просто немає в інших мов;
- Автоматизовано побудова паралельних програм;
- Поєднується строга класика у вивченні основ програмування і конструювання надскладних проектів за допомогою модулів;
- Дії над векторами і матрицями нарівні зі скалярами,

робота з комплексними числами нарівні з речовими;
- Накопичені чисельні бібліотеки (IMSL, NAG, LAPACK, BLAS, Intel MKL), з високопродуктивними обчисленнями (MPI, PVM), з графічними інтерфейсами (Quickwin, FORTRAN/TK).

Найбільш популярні компілятори Фортрану: за стандартом Ф95,

Ф03 - Intel fortran compiler 9,10,11 для багатоядерних комп'ютерів і суперкомпьютеров, вільно розповсюджуваний для Linux; за стандартом Ф90-Ф95 - Fortran Power Station 4.0 та 5, 6 фірми Compaq.

1. Основні елементи мови Fortran.

1.1 Константи

Константа використовуються для представлення постійних значень величин (чисел, логічних значень).

Існує 6 типів констант: цілі, дійсні, комплексні, логічні, символьні та константи подвійної точності.
1). Цілі - це прості цілі числа будь-якого знака. Наприклад: 3; 157. Максимальним числом цілого типу на 16-ти розрядних ЕОМ є число 32767, але вже на сучасних 32-х розрядних ЕОМ це число становить 109.
2). Константи дійсного типу - вони можуть записуватися у двох формах:
а). З фіксованою комою - це числа наступного вигляду: -0.125; 1.7; -167.087.
Спочатку записується знак числа "-" або "+" (його можна опустити), ціла частина числа а потім десяткова крапка і дробова частина числа. Дійсне число записується в пам'яті ЕОМ у наближеному вигляді з точністю до 7-го знака після коми.
б). З плаваючою точкою - застосовуються в основному для запису дуже великих або дуже маленьких чисел для більш наочного і зрозумілого уявлення: 0.25E-3;-1.57E2. При записі дійсного числа з плаваючою точкою на початку вказується знак числа, число (ціле або дійсне), потім показник ступеня - латинська буква Е, за якої без пропусків слідує ціла константа зі знаком чи без знаку. Так, -4.E6 відповідає -4 · 106 (без використання Е це число виглядало б -4000000.). Константи дійсного типу можуть перебувати в діапазоні порядку 1037.
3). Константи комплексного типу - являють собою два дійсних числа, укладені в круглі дужки і розділені комою і мають вигляд: (a_1,a_2). Перша константа представляє дійсну, а друга — уявну частину комплексного числа.

Компоненти a_1 і a_2 повинні мати однакові довжини. Приклади комплексних констант:
(1.35, -0.87) = 1.35-0.87 · j;
(1., .1) = 1. +0.1 · j ;

(9.75, -1.26E2) =9 .75-126 · j;
4). Логічні константи - записуються у вигляді:
. TRUE. та **. FALSE.**
Позначають відповідно істина і хибність. Завжди з двох сторін обмежуються точками.
5). Константи подвійної точності - мають такий же вигляд, як речові константи, що містять показник ступеня. Тільки присутня в показнику літера Е замінюється буквою D. Їх точність подання дійсних чисел в 2 рази вище, а діапазон використовуваних значень має порядок 10307. Приклад:
30D-3 (0.03);-0.003D +2 (-0.3) 1,828 D125
6). Текстові константи - можуть бути представлені в 2-х формах:
а). Стара форма - холерітівський рядок. Являє собою число виведених символів n, ознака константи латинська буква H і самі виводяться символи: nHнабір символів
n - ціла Беззнакова константа в діапазоні від 1 до 255. Приклад:
6HMехмат
б). рядок символів, укладена між двома апострофами
'Мехмат-2011'
'MEXMAT''2012'
Символ апостроф всередині тексту відображається 2-ма апострофами, що йдуть підряд.

1.2 Ідентифікатори.

Ідентифікатори використовуються для позначення зміних, масивів, функцій, процедур і т.і. Це послідовність літер і цифр. Першим символом цієї послідовності повинна бути літера. Максимальна кількість символів 6.

Ідентифікатори не повинні співпадати зі службовими словами.

1.3 Змінні.

Змінна - це величина, яка може в програмі набувати різних значень. Змінні розрізняються по іменах. У Фортран-77 і пізніших версіях ім'я змінної може містити більше 6 символів (до 1320), але тільки перших 6 символів є розпізнаваними.
Приклади імен змінних: X, Y1, beta, mod, ambassaDOrt. Імена змінних у програмі можуть бути набрані прописними або рядковими буквами. В останньому імені розпізнаваними будуть тільки перших 6 символів ambass.
За неявному угодою всі змінні вважаються дійсного типу за винятком тих, ім'я яких починається на одну з букв: I, J, K, L, M, N. Якщо ім'я змінної починається на одну з цих букв, то ця змінна вважається цілого типу і вона може містити тільки ціле число. У всіх інших випадках типи змінних задаються за допомогою операторів опису.

1.4 Оператори опису.

До операторів опису відносять оператори:
REAL - описує змінні і масиви дійсного типу.
Якщо є змінна або масив цілого типу (коли ім'я починається на яку букву з I, J, K, L, M, N), то за допомогою оператора REAL можна перетворити її в змінну дійсного типу.
Приклади:

```
 REAL K, LOG (5)
 K = 2.5
```

INTEGER - описує змінні і масиви цілого типу.
Використовується для перетворення змінної або масиву дійсного типу в змінну або масив цілого типу.
Приклади:

```
INTEGER X, TOK, B (10).
TOK = 4
```

B (1) = 132
OMPLEX - описує змінні і масиви комплексного типу. Всі змінні і масиви комплексного типу обов'язково повинні бути описані в програмі оператором COMPLEX.
Після опису за допомогою оператора COMPLEX змінна будь-якого типу ставати змінної комплексного типу.
Приклади:
COMPLEX Q1, QX2, M12
CHARACTER - описує змінні і масиви текстового типу (рядкові). Всі дані строкового типу повинні бути описані. Якщо довжина строкової змінної не перевищує 4-х байт, то її можна не описувати.
Використовується при безпосередній роботі з текстом для опису змінних як текстових.
Приклад:
CHARACTER * 10 C, C1, D12 * 15
Дана рядок означає, що в програмі змінні C, C1 D12 будуть текстового типу (рядкові), причому змінні C, C1 можуть містити до 10 символів. * 10 - позначає груповий описувач довжини змінних. D12 може містити до 15 символів тексту. * 15 - індивідуальний описувач довжини змінної D12.
LOGICAL - описує змінні логічного типу.
Приклад:
LOGICAL T1, T2
DOUBLE PRECISION або **REAL * 8** - описує змінні подвоєною значності.

1.5 Арифметичні операції.

Операції за пріоритетом:
обчислення значень функцій;
** - Піднесення до степеня;
*, / - Множення і ділення;
+, - - Додавання і віднімання;
() - Черговість виконання арифметичних операцій може задаватися за допомогою дужок;
= - Присвоїти значення.

Вбудовані функції мови FORTRAN

sin x	SIN(x)	e^x	EXP(X)
cos x	COS(x)	ln x	ALOG(X)
tg x	TAN(x)	lg x	ALOG10(X)
arctg x	ATAN(x)	|x|	ABS(X)
$\sqrt{2x}$	SQRT(2.*x)	max(a,b)	AMAX1(A,B)
$\sqrt[3]{x}$	X**(1./3.)	min(a,b,c)	AMIN1(A,B,C)

Попередження: функції *sin, cos*, а також від'ємне число не можна підносити до дійсного степеня.

Для тригонометричних функцій кут вказується в радіанах.

2. Оператори присвоєння та керування

2.1 Оператор присвоєння.

У загальному вигляді:

змінна = вираз

Символ "=" означає операцію присвоїти значення. При схожості з арифметичною операцією "одно" (Y = 48), має істотну відмінність в таких виразах як:
I = I +1 або X = X + DX
Останній запис позначає наступне. Взяти число з клітинки X, додати до нього число, що зберігається в комірці DX і результат записати в клітинку X.
Не можуть знаходитися поряд два знаки арифметичних операцій. Їх необхідно відокремлювати дужками. T * (-2 .* X). Арифметичні операції у виразі виконуються зліва направо з урахуванням пріоритету. При кількох операціях зведенні в ступінь вони виконуються справа наліво. Наприклад: A = B ** C ** 2 буде виконуватися як A = B ** (C ** 2). Під час запису арифметичних операцій кількість відкритих дужок повинна дорівнювати кількості закритих дужок.
Негативне число не може бути піднесено до дійсного степеня. Тому у виразах запису функції sin2x запис в програмі має бути: SIN (X) ** 2, де 2 - цілого типу. Позитивне число може бути піднесено до дійсного степеня. У цьому випадку використовується програма з логарифмічними функціями. Процес зведення в цілу ступінь замінюється перемножуванням, ця функція повинна бути записана так:
X ** 2.5 + Y ** (1. / 3.) - 2 .* SIN (X) ** 2 + EXP (2 .* X).

У наведеному виразі бажано використання даних одного типу.

При арифметичних операціях зі змінними або константами цілого типу, проміжний результат є так само цілим числом, тобто зберігається тільки ціла частина проміжного результату, а дробова частина відкидається. Приклади:

A = 1 / 3 = 0

A = 1. / 3 .= 0.33(3)

I = 199/100 = 1

J = 1. / 3 = 0.33(3) при обчисленні, а при пересиланні результату обчислення в елемент пам'яті ЕОМ дробова частина результату відкидається і J буде дорівнює 0. Якщо вираз для записати як Y ** (1 / 3), то результат такого запису при будь-якому Y дорівнюватиме 1. Оскільки ділення 1 цілого типу на 3 дає результат також цілого типу нуль. Будь-яке число Y, підносити до нульового степеня, дорівнюватиме 1. Правильний запис: Y ** (1. / 3.).

Завдання

1.Визначити результат

1) x^{y^z};

2) $1+x+\frac{x^2}{2!}+\frac{x^3}{3!}$;

3) $1+\cfrac{x}{1+\cfrac{x}{1+x}}$;

4) $3\sin x+4\cos^2 x^2-1$;

5) $(x^y / y^x)^{z(y/x)}$;

6) $(a+b)\sin a+(a-b)\cos b+\frac{a-b}{\sin a+\cos b}$;

7) $\frac{x+1}{x-1}+3{,}6\left\{x-(\sin x+1)^2+x^2(\sin x-1)\right\}$;

8) $-4{,}3-x^{y(6{,}2-y^2)}+1$;

9) $(a-b)/(\dfrac{b}{c+b/(c-d)}+c);$

10) $e^{x}-3\sin x+e^{(x^2-1)};$

2. Визначити порядок виконання операцій:

1) $-A*B/A*A/C*B;$

2) $X**Y**Z**A;$

3. Визначити значення арифметичних виразів:

1) 3/X+X**2, якщо X=2 і має цілий тип,

2) 6.98+X**(-2), якщо X=3 і має цілий тип,

3) X+3-X**3, якщо X=-3 і має цілий тип.

4. Визначити значення:

1) $y=\sin^2 a+\cos(a-3.14)+1;$

2) $x=\dfrac{y}{1+\dfrac{y^2}{1+\dfrac{3y^2}{1+\dfrac{y}{1+y}}}};$

3) $P(x)=((((a_5x+a_4)x+a_3)x+a_2)x+a_1)x+a_0$

Відповіді.

1.

1) X**Y**Z

2) 1+X+X**2/2+X**3/6

3) 1+X/(1+X/(1+X))

4) 3*SIN(X)+4*COS(X**2)**2-1

5) (X**Y/Y**X)**(Z*(Y/X))

6) (A+B)*SIN(A)+(A-B)*COS(B)+(A-B)/(SIN(A)+COS(B))

7) (X+1)/(X-1)+3.6*(X-(SIN(X)+1)**2+X**2*(SIN(X)-1))

8) -4.3-X**(Y*(6.2-Y**2))+1

9) (A-B)/(C+B/(C+B/(C-D)))

10) EXP(X)-3*SIN(X)+EXP(X**2-1)

2.

1) A*B, A*B/A, A*B/A*A, A*B/A*A/C, A*B/A*A/C*B, - A*B/A*A/C*B
2) Z**A,Y**Z**A,X**Y**Z**A

4.

1) Y=SIN(A)**2+COS(A-3.14)+1
2) X=Y/(1+Y**2/(1+3*Y**3/(1+Y/(1+Y))))
3) P(X)=((((A(6)*X+A(5)*X+A(4))*X+A(3))*X+A(2))*X+A(1)

Правила набору тексту програм:

В одному рядку на екрані дисплея можна розмістити 80 символів. Текст програми записується з 7 позиції і по 72.
1 2 3 4 5 | 6 | 7 ... ... 72 | 73
Текст, розташований після 72 позиції, на екрані буде видно, але транслятором сприйматися не буде.
1-5 позиції - розташовуються мітки операторів.
6 позиція - для розміщення символу продовження рядка. Якщо в 6-ій позиції рядка розміщуються символи відмінні від "пропусків" то цей рядок є продовженням попередньої. При цьому в попередньому рядку ніяких знаків переносу не ставиться. Для вказівки рядки продовження найчастіше використовується символ "*". Всього може бути 19 рядків продовження.
Якщо в 1-й позиції рядка розміщується латинська буква С або символ "*" або "!", То це рядок коментарів. Вона служить для внесення пояснюючого тексту в програму. Як пояснюючого тексту може бути будь-яка інформація - вона транслятором не сприймається, а служить для читання програми при наступних переглядах її тексту. Всі оператори в програмі виконуються по черзі зверху вниз. Для зміни черговості застосовуються оператори управління. Цифри, внесені з 1 по 5 позицію, є мітками оператора. Мітка оператора - це число, що містить до 5 символів, які позначають умовний номер оператора. Мітки ставлять не на всі оператори, а тільки на ті, на які будуть посилання в програмі. Призначення міток: мітки дають можливість

звертатися до потрібної рядку програми з будь-якого місця цієї програми.
Правило:
В одній програмі не може бути двох однакових позначок. Номери міток ставляться в довільному порядку. Обов'язково повинні бути мітки після операторів GOTO і IF арифметичного.

Скласти програму для обчислення площі трикутника за формулою Герона

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

$$p = (a+b+c)/2$$

В якості вихідних даних є значення сторін трикутника a, b, c які при виконанні програми потрібно буде ввести з екрану дисплея як три дійсних числа. При введенні дані відокремлюються один від одного комами або символами пробіл.
У процесі роботи програми необхідно обчислити значення площі S і вивести це значення на екран.

Програма № 1 - Обчислення площі трикутника

```
4     WRITE (6, *) 'Введіть значення сторін трикутника A, B, C'
```

*Оператор WRITE служить для виведення інформації. В даному випадку на екран дисплея (тому варто цифра 6). Символ * означає, що висновок бесформатний (спрощена) При виконанні цього оператора на екрані дисплея з'явиться запрошення до введення інформації: 'Введіть значення сторін трикутника A, B, C' (бажано всі програми постачати такий рядком).*

```
      READ (5, *) A, B, C
```

READ служить для введення інформації. A, B, C - список імен змінних які необхідно запровадити. 5 - канал екрана дисплея. Символ "" - безформатний спрощений висновок (введення).*
Цей рядок розшифровується так: ввести з екрану дисплея числові дані для змінних A, B, C.

P = (A + B + C) / 2.

Скласти значення змінних, що зберігаються в змінних A, B, C, розділити на 2 і результат записати в змінну P.

S = SQRT (P * (PA) * (PB) * (PC))

Обчислюється значення змінної S.

WRITE (6, *) 'Площа трикутника зі сторонами' A, B, C, 'дорівнює', S

Виводиться інформація, що знаходиться в списку виведення за дужками оператора WRITE. На екрані з'явиться текст Площа трикутника зі сторонами, потім чисельні значення змінних A, B, C, потім текст дорівнює та чисельне значення змінної S.

GO TO 4 *Цей оператор здійснює перехід на мітку 4 в початок програми. Таким чином зациклюється введення нових вихідних даних для обчислення площі іншого трикутника.*

6 STOP

END *Оператори STOP і END здійснюють стандартне завершення програми. Оскільки оператор STOP розташований після GO TO, то він повинен мати мітку (хоча в програмі на мітку 6 і немає посилань).*

2.2Арифметичний оператор IF

Служить для розгалуження програм в 3-х або 2-х напрямках залежно від заданих умов. Записується у вигляді:
IF (арифметичне вираз) m1, m2, m3
де m1, m2, m3 - мітки операторів, на які буде передаватися управління обчислювальним процесом.
Приклад: IF (2 .* A-SIN (B)) 3,4,12
IF арифметичний працює наступним чином:
1). Обчислюється арифметичне вираження в дужках.
2). Обчислення значення порівнюється з нулем.
3). Якщо обчислене значення <0, то управління передається на мітку m1 (3), якщо обчислене значення = 0, то управління передається на мітку m2 (4), якщо> 0, то на мітку m3 (12).

За допомогою IF арифметичного, приміром, можна перевіряти Підкореневий вираз і якщо воно виявиться негативним (не можна добути корінь квадратний з негативного числа за законами математики), то управління передасться на потрібну позначку і не відбудеться переривання виконання програми при спробі обчислення кореня з від'ємного числа.

Примітка:

У IF арифметичному повинно бути завжди три мітки, дві з них можуть бути однаковими. Мітки можуть бути розташовані в будь-якому місці програми (вище або нижче оператора IF).

Правило:

1. Оператор, наступний після IF арифметичного повинен мати позначку;
2. IF арифметичний не може бути останнім оператором в циклі DO.

Якщо в якості умови є нерівність, то його необхідно привести до виду, при якому відбувається порівнювання висловлювання з нулем.

x2 + c> y → x2 + c-y> 0.

З урахуванням цього допрацюємо Програму № 1, де за допомогою IF арифметичного перевіримо умова: якщо найдовша сторона більше ніж напівпериметр, то трикутник не існує.

Програма № 2 - Обчислення площі трикутника за допомогою IF арифметичного.

```
4     WRITE (6, *) 'Введіть значення сторін трикутника A, B, C'
      READ (5, *) A, B, C
C     Блок перевірки правильності введених даних
      IF (A) 4,4,20 Перевірка введених вихідних даних:
сторона трикутника не може бути негативною або
дорівнює нулю.
20    IF (B) 4,4,21
21    IF (C) 4,4,22
```

```
*Замість цих трьох операторів краще використовувати
один IF (AMIN1 (A, B, C)) 4,4,22*
C     *Кінець блоку перевірки правильності введених даних*
22    P = (A + B + C) / 2.
      IF (AMAX1 (A, B, C)-P) 6,8,9
C     *Вбудовані функції AMIN1, AMAX1 зі списку змінних,
перерахованих в дужках, вибирають мінімальне
(максимальне) число. Останній оператор позначає: якщо
максимальна зі сторін дорівнює напівпериметр, то
управління передається на мітку 8. Якщо максимальна зі
сторін менше напівпериметр, то йдемо на мітку 6 і
обчислюємо площу. А інакше йдемо на мітку 9.*
8     WRITE (6, *) 'Площа трикутника дорівнює нулю'
      GO TO 4
6     S = SQRT (P * (P-A) * (P-B) * (P-C))
      WRITE (6, *) 'Площа S =', S
      PAUSE
      GO TO 4
9     WRITE (6, *) 'Такий трикутник не існує'
C     Оператор PAUSE або PAUSE 'текст' служить для
призупинення виконання програми на екрані до будь-якого
натискання на клавіатуру. Він дозволяє подивитися дані,
отримані при виконанні програми до завершення програми
по оператору STOP.
      STOP
      END
```

Опис роботи програми:

Спочатку виводиться запрошення до введення сторін трикутника A, B, C. Після їх введення з клавіатури відбувається привласнення цих чисел змінним A, B, C відповідно. Потім йде блок перевірки, в якому перевіряється, щоб не було введено негативне число (якщо введено негативне число, то управління передасться на мітку 4 - мітку оператора запрошення до введення). Після блоку перевірки обчислюється значення напівпериметр P. Перевірка на коректність введених даних і можливість

існування трикутника з такими сторонами здійснюється за допомогою умови: IF (AMAX1 (A, B, C)-P) 6,8,9.

2.3 IF логічний

Записується у вигляді:
IF (логічне вираз) оператор, що виконується
приклад:
IF (A.GT.B) Y = SIN (X) Якщо А більше В той Y присвоїти значення sin (x)
В якості виконуваних операторів можуть бути:
1). Оператори присвоювання типу X = 3.5;
2). Оператори введення - виведення інформації READ, WRITE;
3). Оператор переходу GO TO;
4). Оператор виклику підпрограми CALL;
5). Оператор PAUSE.
У логічних виразах відбувається порівняння значень двох виразів або змінних. Ці висловлювання поділяються операцією відносини.
Операції відносини відповідають математичним позначеннями:

.GT. >
.GE. ≥
.EQ. =
.NE. ≠
.LE. ≤
.LT. <

Розглянемо, що означає вираз:
IF (A.GT.B) Y = SIN (X)
Це означає: якщо A більше B, то Y присвоюється синус від X. Потім після цього оператора буде виконуватися наступний за ним. Якщо A не більше B, то оператор присвоювання Y = ігнорується і виконується оператор, наступний після IF.
Крім операцій відносини використовують, і логічні оператори, за допомогою яких можна організувати більш складні умови.
Логічні оператори:

. AND. – Логічне 'І'
. OR. - Логічне 'АБО'
. NOT. - Логічне 'НІ'

Приклад:
якщо a + b ≥ c> 2 * sin x, то надрукувати a, b, x
IF (A + B.GE.C.AND.C.GT.2 .* SIN (X)) WRITE (6, *) A, B, X
IF логічний працює наступним чином: Визначається логічне вираз, що стоїть в дужках. Це логічне вираз може бути істинно і мати значення. TRUE. або може бути помилковим і мати логічне значення. FALSE .. Якщо логічний вираз в дужках істинно, то виконується виконується оператор, що стоїть за дужками оператора IF. Якщо логічний вираз в дужках хибне, то виконується оператор не виконується. Як у першому, так і в другому випадку після IF логічного виконується наступний за ним по тексту програми оператор, за винятком випадку, коли логічний вираз істинно, а виконуваних оператором є GO TO. (Приклад: IF (A.GT.0.) GO TO 7).

Завдання: Обчислити значення λ в залежності від даних

умов.

$$\lambda = \left\{ \begin{matrix} \sin x \\ 0 \\ \cos x \end{matrix} \right\} \quad якщо \quad \begin{matrix} x > 2b \\ x = 2b \\ x < 2b \end{matrix}$$

Перетворимо нерівність x> 2b до виду порівняння з нулем x - 2b> 0

x = 2b -> x - 2b = 0

x <2b -> x - 2b <0

Програма № 3 з IF арифметичним

```
REAL LAM
```

У цьому рядку мінлива LAM описується як змінна дійсного типу.

Якщо не зробити цього опису, а використовувати в програмі змінну LAM, то через невідповідність типів (LAM - цілого типу, а використовується як речового) буде виникати помилка. У цієї змінної не може бути дробової частини.

```
WRITE (6, *) 'Введіть значення x, b'
READ (5, *) X, B
IF (X-2. * B) 4,7,8 Порівнюємо умову і переходимо на одну з міток
8 LAM = SIN (X)
GO TO 10
7 LAM = 0.
GO TO 10
4 LAM = COS (X)
10 WRITE (6, *) 'Обчислення значення Лямбда =', LAM
PAUSE
STOP
END
```

70% помилок у програмістів виникає через невідповідність типів змінних!

Помилки ніби і несуттєві, але через них може не запрацювати навіть абсолютно правильно складена

програма. І на усунення помилки може піти досить багато часу. Зверніть увагу!
Як і всі програми, ця програма може бути складена великою кількістю різних способів. Наведемо ще один спосіб вирішення цієї ж задачі.

Програма № 4 з використанням IF логічного
REAL LAM *Замість опису імена LAM як дійсного типу можна задати інше ім'я, яке не буде починатися на букви I, J, K, L, M, N, наприклад ALAM*

```
WRITE (6, *) 'Введіть значення x, b'
READ (5, *) X, B
IF (X.GT.2 .* B) LAM = SIN (X)
IF (X.EQ.2 .* B) LAM = 0.
IF (X.LT.2 .* B) LAM = COS (X)
WRITE (6, *) 'Обчислення значення Лямбда =', LAM
PAUSE
STOP
END
```

Краще ще ввести нову змінну і привласнити їй вираз 2 .* B і далі використовувати цю змінну:

```
... ... ....
A = 2 .* B
IF (X.GT.A) LAM = SIN (X)
```

До речі, замість першого оператора IF (X.GT.2 .* B) LAM = SIN (X) можна просто записати LAM = SIN (X). Проаналізуйте програму і переконайтеся, що вона буде працювати правильно. Проаналізуйте, що станеться, якщо таким же чином замінити останній оператор IF.

2.4 Табулювання функції.

Табулювання функції - це обчислення будь-якої функції з аргументом, що змінюється в якихось межах.
Завдання:
Обчислити і надрукувати всі значення функції:

$$f(x) = \sin x^2 - e^x \cos 2x,$$

якщо x змінюється в межах від a до b з кроком dx.

```
C     Програма № 5. Цикли з оператором IF.
      WRITE (6, *) 'Введіть Хнач, Хкон, крок циклу'
      READ (5, *) XN, XK, DX
C     Блок коригування кроку
      N = (XK-XN) / DX +1 визначаємо кількість обчислень на відрізку a ÷ b
      DX = (XK-XN) / (N-1) уточнюємо значення кроку
C     Змінній циклу присвоюємо початкове значення
      X = XN
C     Обчислюємо значення функції
11    F = SIN (X * X)-EXP (X) * COS (X) ** 2
У показника ступеня 2 крапку ставити не можна, тому що не можна звести негативне число в речову ступінь.
C     Виводимо результат обчислення на екран
      WRITE (6, *) 'При X =', X, 'значення функції дорівнює', F
Виводимо на екран значення аргументу X і відповідне йому значення функції F.
C     Задаємо приріщення аргументу
      X = X + DX
C     Перевіряємо умову повторення циклу
      IF (X.LE.XK) GO TO 11
      PAUSE
      STOP
      END
```

Для організації циклу за допомогою оператора IF ми зробили:

1). X присвоїли XN:

```
X = XN
```

2). Зробили обчислення значення F в області циклу:

```
F = SIN (X * X)-EXP (X) * COS (X) ** 2
```

3). Збільшили значення аргументу на величину кроку циклу: X = X + DX

4). За допомогою IF логічного перевірили умова повторення циклу:

IF (X.LE.XK) GO TO 11.

При обчисленні значень цієї функції, наприклад, при X, змінюється від 1 до 10 з кроком 2, поточне значення X дорівнюватиме 1,3,5,7,9. Тобто не буде отримано останнє значення функції на заданому відрізку. Тому в програмі і застосовується блок коригування кроку. Кількість обчислень на відрізку буде N = (10-1) / 2 +1 = 5 (ціле число). Нове, уточнене значення DX дорівнюватиме (10-1) / (5-1) = 2.25

2.5 Оператор циклу DO

Оператор **DO** призначений для створення циклів, що забезпечують багаторазове виконання групи операторів в програмі.

Оператор **DO** в порівнянні с **IF** є більш потужним інструментом для організації циклів. Форма запису наступна:

DO m i = i1, i2, i3,

де **m** - мітка оператора, який є останнім в області циклу **DO**; **i** - змінна циклу, що змінюється від початкового значення **i1** до кінцевого значення **i2** з кроком **i3**.

Приклад:

```
      DO 8 I=1,13,2
      M=I*I                  область циклу
    8 WRITE(6,*) I,M
      . . .
```

Оператор циклу **DO** працює наступним чином: Змінної циклу **i** присвоюється початкове значення **i1**, потім виконуються всі оператори в області циклу, включаючи і

останній з міткою **m**. Потім до величини змінної і додається величина кроку **i3** і отримане нове значення змінної циклу порівнюється з кінцевим значенням **i2**. Якщо нове значення змінної циклу більше ніж значення **i2**, то здійснюється вихід з циклу і виконання передається оператору, наступному після мітки m. Якщо значення змінної циклу і менше, ніж **i2**, то повторюється обчислення в області циклу, починаючи з оператора, наступного після **DO**.

Після закінчення циклу змінна циклу зберігає завжди значення більше, ніж кінцеве значення циклу i2. У разі виходу з циклу до його завершення змінна збереже своє поточне значення.

В якості оператора, який є останнім у циклі DO, краще ставити оператор CONTINUE, який не виконує ніяких дій, а просто показує, що на ньому закінчився цикл. Оператор CONTINUE найчастіше є останнім оператором циклу DO в тих випадках, коли останнім оператором циклу DO може опинитися один з операторів: **IF (арифметичний), PAUSE, STOP або інший керуючий оператор.**
Приклад:

```
      DO 4 I=1,8,2
      X=X+2.
      . . .
      IF(X) 4,2,3
```

Оскільки останнім оператором тіла DO не може бути арифметичний оператор IF

```
4     CONTINUE
      . . .
```

Якщо крок циклу зробити негативним, а початкове і кінцеве значення позитивним, то відбудеться зациклення програми до переповнення розрядної сітки ЕОМ (-32 767). Починаючи з версії ФОРТРАН-77 в якості змінної циклу можна використовувати змінні дійсного типу.

Наприклад:

```
      DO 7 X=XN , XK , DX
           ...
7     WRITE(*,*) X , Y
```

Оператор циклу DO не передбачає коригування кроку DX.

Для заданого n обчислити **n!** (факториал от n) та обчислити сумму квадратів, за допомогою оператора циклу DO.

```
C     Програма №6. Обчислення суми та факторіалу від N
      WRITE(6,*)'Введіть N'
      READ(5,*) N
C     Обчислення суми N
      S=0.
      DO 2  I=1,N
      S=S+I**2
2     CONTINUE
C     Обчислення факторіалу від N
      P=1.
      DO 3  I=1,N
3     P=P*I
      WRITE(6,*)'Сума S=',S,'  факторіал =',P
      PAUSE
      STOP
      END
```

Завдання

2. Скласти програму обчислення степенів:

1) $y = x^n, n -$ натуральне число

2) $y = \frac{1}{x^n}, n -$ натуральне число

3) $y = x^n, n -$ ціле число.

3. Скласти програму обчислення

1) $y=sin(sin(...sin(x)...))$ (n раз)*;*

2) $y=sin^n x$.

4. Скласти програму обчислення добутку

$a*b*a*b*...*a*b*a$ ($2n$ знаків множення).

5.Скласти програму для обчислення значень многочленів і виконати їх при заданих значеннях аргументів:

1) $y = x^n + x^{n-1} + ... + x^2 + x + 1$, n=3, x=2;

2) $y = x^{2n} + x^{2n-1} + ... + x^4 + x^2 + 1$, n=4,x=1;

6. Скласти алгоритм обчислення добутку $p=m^n$, використовуючи операцію додавання та виконати його при m=5, n=3.

7. Скласти алгоритми обчислення значень многочлена

а) $y = nx^{n-1}+(n-1)x^{n-2}+...+2x+1$;

б) $y = x^n(1-x)^m$, $(m,n \neq 0)$;

Завдання:

Знайти всі значення функції

$$\mu = \sqrt{\frac{a-b}{c-a}} * sin^2(\sqrt[3]{|x-1|} + e^{2x}$$

якщо х змінюється в межах:

$X=X_H \div X_K$ с кроком dx.

Визначити максимальне і мінімальне значення.

Якщо значення $\mu<0$, то результат вивести в лівій частині екрана, а якщо $\mu>0$ - у правій частині.

```
C       Програма № 7. Максимум і мінімум функції
3       WRITE (6, *) 'Введіть змінні a, b, c'
```

```
      READ (5, *) A, B, C
      WRITE (6, *) 'Введіть значення Хнач, Хкон, крок dx'
      READ (5, *) XN, XK, XD
C     Блок коригування кроку
      N = (XK-XN) / XD +1.
      DX = (XK-XN) / (N-1)
      P = (A-B) / (C-A)
      IF (P.LT.0.) GO TO 3 Перевірка на мінус під коренем
      FMA =- 1.E30 початкове значення змінної, де буде
      запам'ятовуватися максимум, задаємо мале число,
      розташоване       ліворуч на числової осі.
      FMI = 1.E30 А для мінімуму задаємо початкове велике
число.
      DO 4 X = XN, XK, DX
      S = SIN (ABS (X-1.) ** (1. / 3.) + EXP (2 .* X)) ** 2
      F = SQRT (P * S)
      IF (F.GT.FMA) FMA = F або можна так: FMA =
AMAX1 (F,    FMA)
      FMI = AMIN1 (FMI, F) або IF (F.LT.FMI) FMI = F
      IF (F.LT.0.) WRITE (6, *) 'При x =', X, 'MU =', F
4     IF (F.GT.0.) WRITE (6, *) 'При x =', X, 'MU =', F
      PAUSE
      STOP
      END
```

3. Обчислення логічних виразів

Порядок виконання операцій в логічних виразах визначається їх пріоритетом. В таблиці, що наведена нижче:
− L1, L2 – логічні вирази;
− a, b – числові або текстові вирази.
В відношеннях можна порівнювати не тільки числа, але й рядки – вони порівнюються у відповідності до алфавіту.

Операція		Фортран-77	Фортран-90	Пріоритет
Обчислювання значень a,b		Числові або символьні операції.		1
Операції відношення	Більше	a.GT.b	a > b	2
	Більше або дорівнює	a.GE.b	a >= b	2
	Менше	a.LT.b	a < b	2
	Менше або дорівнює	a.LE.b	a <= b	2
	Дорівнює	a.EQ.b	a = = b	2
	Не дорівнює	a.NE.b	a /= b	2
Логічні операції	Заперечення	.NOT.L1		3
	Конюнкція	L2.AND.L1		4
	Дизюнкція	L2.OR.L1		5
	Еквівалентність	L2.EQV.L1		6
	Нееквівалентність	L2.NEQV.L1		6

Позначення в логічній формулі:

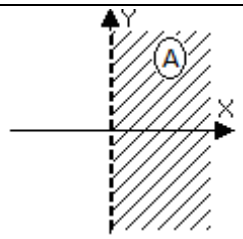
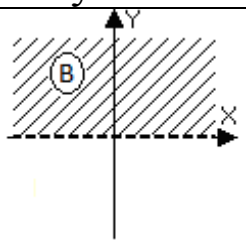
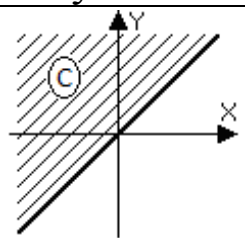
$\overline{B \bullet C}$ – заперечення логічного добутку *B* та *C*,
$\overline{B + C}$ – заперечення логічної суми *B* та *C*.

Завдання:
Завдання містить в собі три нерівності, наприклад,
x > 0 (A) y > 0 (*B*) y ≥ x (C)

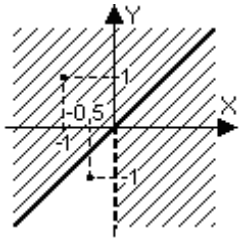
Нерівності A, B, C входят в логические формулы $A+B+C,\ A\bullet B\bullet C$, загальні для всіх варіантів, і в додаткову логічну формулу, наприклад, $\overline{A\bullet B}+C.$

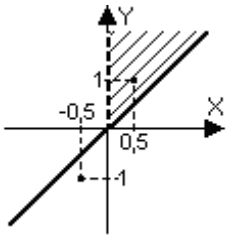
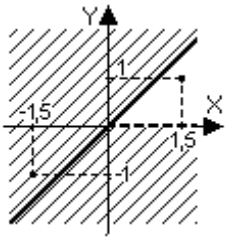
Підготовка проходить в три етапи:

1. Графічна інтерпретація початкових нерівностей и запис на Фортрані відповідних відношень.

x > 0	y > 0	y ≥ x
		
x>0	y>0	y>=0

2. Системи нерівностей, їх графічна інтерпретація, логічні вирази, відповідні системам нерівностей

Операція	Система нерівностей	Графічна інтерпретація	Контрольні точки	Логічний вираз
Логічне додавання	$A+B+C,$ $\begin{bmatrix} x>0 \\ y>0 \\ y\geq x \end{bmatrix}$		TRUE (-1, 1); FALSE (-0.5, -1)	x>0. **.OR.** y>0. **.OR.** y>=x Істина для точок в I або II або IV квадрантах або вище прямої $y = x$

Логічний добуток	$A \bullet B \bullet C,$ $\begin{cases} x > 0 \\ y > 0 \\ y \geq x \end{cases}$		TRUE (0.5, 1); FALSE (1, -1)	x>0. **.AND.** y>0. **.AND.**y>=x Істина для точок в I квадранті та вище прямої *y = x*
Додаткова формула	$\overline{A \bullet B} + C,$ $\left[\begin{cases} x > 0 \\ y > 0 \end{cases}\right.$ $\left. y \geq x \right.$		TRUE (-1.5, -1); FALSE (1.5, 1)	.**NOT**. (x>0. **.AND.** y>0.) **.OR.** y>=x Істина для всіх точок в II, III або IV квадрантах або вище прямої *y = x*

Програма

! Приклад програми для однієї контрольної точки:
Program Logical_expressions
Implicit None
Logical A, B, C, ILI *! ILI – результат для операції OR*
REAL x, y *! координаты точек*
Character*6 prognos *! рядок довжиною 6 символів*
Open(1, file='In.txt') *! файл з початковими даними*
Open(6, file='Out.txt') *! файл результатів*
*!******
! операція OR: для об'єднання областей
READ(1,*) x, y, prognos *! вводяться координати однієї точки і прогноз*
! логічні оператори присвоєння:
A = x>0; B = y>0; C = y>x ! *операції відношення*
ILI = A **.OR.** B **.OR. C** ! *логічна операція*
Write(6,*)'для OR ', prognos, ' (', x, ',', y, ') відповідь ', ILI
*!******

*! Для кожної з п'яти контрольних точок, що залишилися, додати оператори, аналогічні частині програми від «!*****» до «!*****»*

End Program Logical_expressions

Варіанти завдань

№	A	B	C	Додаткова логічна формула
1	$x>0$	$y>0$	$y>x^2-2$	$A+\overline{B+C}$
2	$x\le 0$	$y>0$	$y\le -\frac{x^2}{3}+3$	$\overline{A\bullet B}+C$
3	$x>0$	$y\le 0$	$x^2+y^2<1$	$\overline{A\bullet B}+C$
4	$x>0$	$y>0$	$\frac{y^2}{3}-3\le x$	$\overline{A\bullet B}+C$
5	$x\le 0$	$y>0$	$x^2+y^2\le 1$	$\overline{A\bullet B}+C$
6	$x\le 0$	$y>0$	$y>x^2-2$	$\overline{A\bullet B}+C$
7	$x\le 0$	$y>0$	$y>-x^2+2$	$\overline{A\bullet B}+C$
8	$x\le 0$	$y>0$	$\frac{x^2}{4}+y^2\le 1$	$\overline{A\bullet B}+C$
9	$x>0$	$y>0$	$\frac{y^2}{4}+x^2\le 1$	$\overline{A\bullet B}+C$
10	$x\le 0$	$y>0$	$\frac{y^2}{2}\le x^2+1$	$\overline{A\bullet B}+C$
11	$x>0$	$y>0$	$y\le x^2-2$	$\overline{A\bullet B}+C$
12	$x\le 0$	$y\le 0$	$y\le -x^2+2$	$C\bullet\overline{A+B}$
№	A	B	C	Додаткова логічна формула
13	$x>0$	$y\le 0$	$\frac{x^2}{4}+y^2\le 1$	$C\bullet\overline{A+B}$
14	$x>0$	$y\le 0$	$\frac{y^2}{4}+x^2>1$	$C\bullet\overline{A+B}$
15	$x>0$	$y>0$	$\frac{y^2}{2}>x+1$	$C\bullet\overline{A+B}$

16	$x \le 0$	$y \le 0$	$x^2 + y^2 > 1$	$A + \overline{B + C}$
17	$x \le 0$	$y > 0$	$y \le -\frac{x^2}{3} + 3$	$A + \overline{B + C}$
18	$x > 0$	$y > 0$	$y > -x^2 + 2$	$A + \overline{B + C}$
19	$x \le 0$	$y \le 0$	$\frac{y^2}{3} - 3 \le x$	$A + \overline{B + C}$
20	$x > 0$	$y > 0$	$x^2 + y^2 > 1$	$A + \overline{B + C}$
21	$x > 0$	$y \le 0$	$y > x^2 - 2$	$A + \overline{B + C}$
22	$x > 0$	$y > 0$	$y \le -x^2 + 2$	$B \bullet \overline{A + C}$
23	$x \le 0$	$y \le 0$	$\frac{x^2}{4} + y^2 \le 1$	$B \bullet \overline{A + C}$
24	$x \le 0$	$y \le 0$	$\frac{y^2}{4} + x^2 \le 1$	$B \bullet \overline{A + C}$

4. Масиви.

Під масивом розуміють сукупність даних одного типу, об'єднаних одним ім'ям. Масиви повинні бути описані в програмі одним з операторів опису. Існує спеціальний оператор для опису масивів:

DIMENSION

Приклад:

DIMENSION A (10), B (8), C (3,4), K (15).

Оператор не виконує жодних дій у програмі і служить лише тільки для відведення місця в оперативній пам'яті ЕОМ для розміщення масивів.

Спочатку всіх елементів масиву задані значення, рівні нулю, крім ЕОМ зразка до 1985, де масив спочатку був заповнений машинним нулем (0.67 * 10-19).

Масиви бувають:

а). Одновимірними (інша їх назва - вектор-стовпець);

б). Двомірними (інша назва - матриця)

в). Трьох-і т.п. мірними. Максимальна кількість вимірювань в масиві дорівнює 7.

Щоб звернутися до певного елемента масиву, необхідно вказати його ім'я та індекс цього елемента. Індекс у одновимірному масиві - це порядковий номер елемента масиву, для двомірного - номер рядка та номер стовпця.

В якості індексу може виступати константа цілого типу, мінлива цілого типу або арифметичне вираз, бажано цілого типу.

Приклад:

```
DIMENSION A(10)
    K=4
    M=3
    A(1)=2.5
    A(3)=1.7E-7
    A(K+M)=D+A(1)
    A(4)=A(M)+A(K+1)
```

Цей рядок аналогічний виразу:

A(4)=A(3)+A(4+1)

змінна цілого типу — *арифметичний вираз*

DIMENSION A (10)
K = 4
M = 3
A (1) = 2.5
A (3) = 1.7E-7
A (K + M) = D + A (1)
A (4) = A (M) + A (K +1) *Цей рядок аналогічний :*
A (4) = A (3) + A (4 +1)

Примітка:
Бажано уникати арифметичних дій з індексами масиву, особливо при циклічному обчисленні, тому що це дуже уповільнює виконання програми.
Двомірні масиви в пам'яті ЕОМ розташовуються по стовпцях:

	1	2	3	4
1		1.8		
2				
3				

Приклад заповнення клітинки двомірного масиву: C (1,2) = 1.8
При цьому в рядок номер 1 і стовпець номер 2 масиву З заноситься число 1.8.
При виконанні програми транслятор аналізує, чи не перевищує значення викликається індексу масиву його граничного значення, описаного в операторі DIMENSION (тобто перевіряється випадок, коли викликається індекс масиву не існує - масив має менші розміри). Так при записі C (K, M) = C (1,1) + C (1,2), яка відповідає запису: C (4,3) = C (1,1) + C (1,2) значення C (4,3) не відповідає. При спробі виконати таку операцію буде видана

помилка.
При роботі з двомірними масивами є одна особливість, яку розглянемо на прикладі:
READ (5, *) До
С (К, 3) = 7
Поки змінну До будемо задавати рівну від 1 до 3, програма буде виконуватися коректно. Але, якщо До присвоїти значення 4 і більше, то ЕОМ буде не в змозі відстежити, що межі розмірності масиву порушені. Звернення буде йти до зовсім іншого, і навіть випадковому елементу пам'яті ЕОМ. Масиви можна описати ще й за допомогою операторів опису **REAL і INTEGER:**
REAL A (10)
INTEGER K (15)
Повторне опис масивів не допускається.
Масиви можуть бути: цілого, речового, комплексного, строкового і логічного типу.
Для 8-розрядних ЕОМ максимальне значення індексу масиву може бути одно 32767. Наприклад: DIMENSION A (32767), B (32767, 32767).

Завдання:

Обчислити значення певного інтеграла на відрізку від a до b за методом прямокутників

$$\lambda = \int_a^b (sin^2 x + cos x^2)dx$$

Замінимо обчислювання интеграла обчислюванням суми:

$$\int_a^b f(x)dx = \sum_{i=1}^{n-1} f(x)dx$$

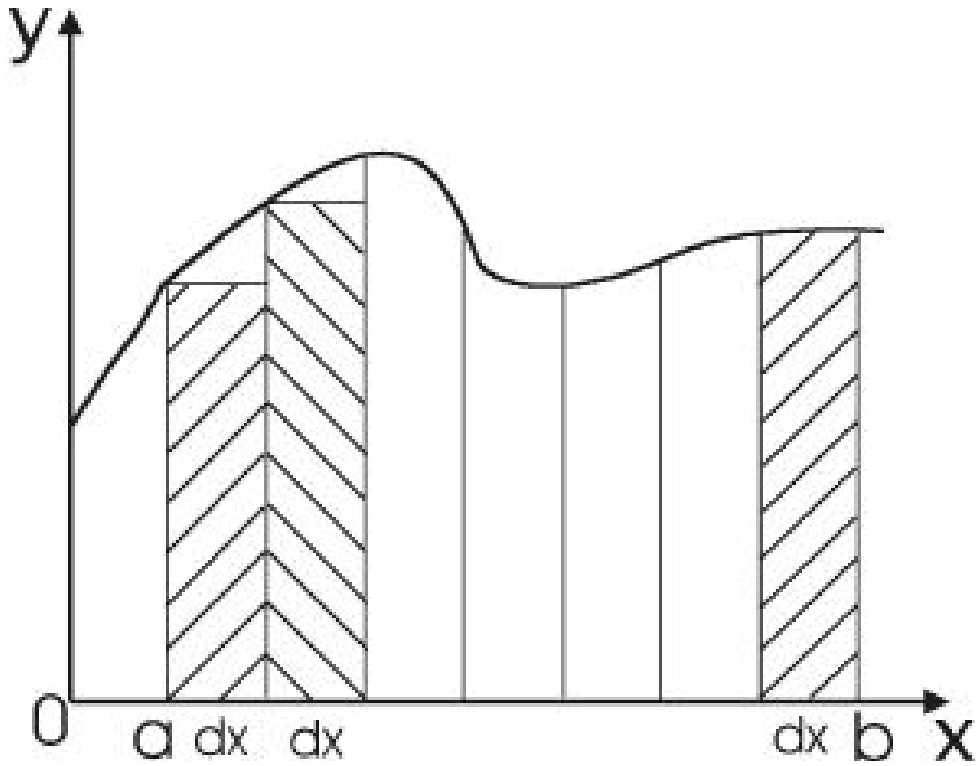


Рис.1. Схема дискретизації за методом прямокутників.

Програма № 8.

```
C     Обчислення визначеного інтеграла за методом
      прямокутників
1     WRITE (6, *) 'Введіть значення A, B, DX'
      READ (5, *) A, B, DX
C     Обчислюємо кількість точок на кривій
      N = (B-A) / DX +1
      DX = (B-A) / (N-1)
C     Обчислення суми
      F = 0.
```

Як завжди, перед обчисленням суми треба занулити елемент пам'яті

```
      DO 2 X = A, B-DX, DX
```

Кінцеве значення циклу беремо рівним B-DX для того, щоб не обчислити велику площу, тому що після закінчення циклу змінна циклу має значення, яке більше кінцевого значення циклу на величину кроку DX.

```
      FX = SIN (X) ** 2 + COS (X ** 2)
      2 F = F + FX * DX
      WRITE (6, *) 'Значення інтеграла дорівнює', F
      PAUSE
      GO TO 1
```

```
STOP
END
```

Оператором циклу DO змінної циклу X присвоюється початкове значення A. Організовуються циклічні обчислення в області циклу, включаючи оператор з міткою 2. На кожному новому циклі значення X збільшується на величину кроку DX. В області циклу спочатку обчислюється значення функції FX, а потім до змінної F, в якій накопичується сума значень локальних інтегралів, додається значення функції, помножене на DX. Обчислення будуть проводитися до досягнення змінної циклу X значення B-DX. Минулий раз до F додасться FX * DX (остання площадка) і в галузі циклу обчислення завершаться.
Чим менше величина кроку DX, тим вище точність обчислення інтеграла.

Завдання 2:

Обчислити значення попереднього інтеграла, методом трапеції, який більш точний.

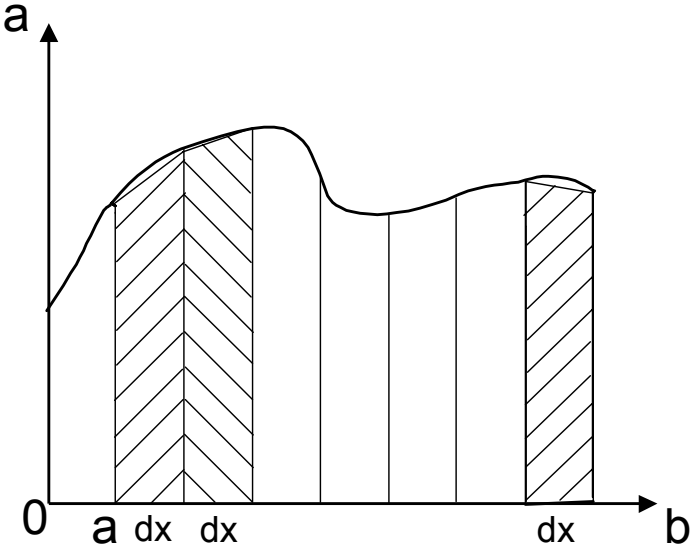


Рис.2. Схема дискретизації за методом трапецій

$$\int_a^b f(x)dx = \sum_{i=1}^{n-1} \frac{f(i)+f(i+1)}{2} dx$$

Розв'язок.

1. Обчислимо всі значення функції F (x) і запам'ятаємо їх у масиві.
2. Роздрукуємо всі значення функції при відповідному аргументі.
3. Обчислимо значення суми для отримання значення інтеграла за методом трапеції.

Програма № 9

```
C     Обчислення визначеного інтеграла за методом трапеції
      DIMENSION FX (1000)
Задаємо розмірність масиву завідомо більшу, ніж можлива кількість обчислень функції
      WRITE (6, *) 'Введіть значення величин A, B, DX'
      READ (5, *) A, B, DX
      N = (BA) / DX +1 Кількість обчислень на відрізку
      DX = (BA) / (N-1) Уточнене значення кроку
C     Обчислюємо значення масиву
      X = A Задаємо початкове значення аргументу X
      DO 2 I = 1, N
      FX (I) = SIN (X) ** 2 + COS (X ** 2) Формуємо масив значень функції
      WRITE (6, *) 'При X =', X, 'значення функції одне", FX (I)
    2 X = X + DX Задаємо приросщення аргументу
C     Обчислення інтеграла за формулою трапеції
      F = 0. Занулити елемент пам'яті для накопичення суми
      DO 1 K = 1, N-1 Зверніть увагу на кількість циклів
      F = F + (FX (K) + FX (K +1)) / 2 .* DX
1     CONTINUE
      WRITE (6, *) 'Значення інтеграла дорівнює', F
      PAUSE
      STOP
      END
```

Робота з одновимірними масивами.

Введення масивів.

Припустимо, що в програмі описані масиви:
DIMENSION A(10),B(10),C(20),D(4,3)
Існує декілька засобів введення масивів

1) **READ(5,*) A**

Означає ввести всі числові значення масиву А в тій кількості, скільки їх описано в операторі DIMENSION.

2) **READ(5,*) (A(I),I=1,10)**

Тут введення масиву здійснюється за допомогою неявного циклу DO, де I змінюється від 1 до 10 з кроком 1.

3) **READ(5,*) N**
READ(5,*) (A(I),I=1,N)
або: **READ(5,*) N,(A(I),I=1,N)**

Спочатку вводиться число N, що позначає кількість вводяться елементів масиву, а потім вводиться N-ну кількість елементів масиву.

Виведення масивів.

1) **WRITE(6,*) A**

При спрощеному виведенні масиву А на екран дисплея виведуться значення всіх елементів масиву, кількість яких була описана в операторі DIMENSION. У кожному рядку друкується по 5 чисел у вигляді:
-0.1234567E-01
тобто з точністю до 7 знаків після коми.

2) **WRITE(6,*) (A(I),I=1,N) ,** *де N – задано*

3) **WRITE(6,*) ('A(',I,')=',A(I),I=1,N)**

Використовується неявний цикл типу DO, по якому відбувається виведення імені масиву за допомогою константи типу рядок, а за ним в дужках вказаний номер елемента, що виводиться, і після '=' числове значення элементу масиву: A(…1) = число A(…2) = число ...

4) **DO 18 I=1,N**

18 WRITE(6,*) 'A(',I,')=',A(I)

Використовується зовнішній цикл DO.

Виведення масиву в два стовбчики:

```
      DO 28 I=1,N,2
   28 WRITE(6,*) 'A(',I,')=',A(I),'
A(',I+1,')=',A(I+1)
```

Завдання:

Знайти суму елементів одновимірного масиву.

Фрагмент програми:

```
      S=0.
      DO 30 I=1,N
   30 S=S+A(I)
```

Завдання:

Визначити значення μ при $\alpha=t_н \div t_k$ з кроком dt.

$$\mu = arctg \sqrt[3]{\left|\frac{\ln \alpha}{e^{2\alpha}}\right|}$$

```
      REAL MU(1000)
      Оскільки назва масиву починається з літери
      M,товона буде сприйнята, як змінна цілого типу.
      Тому необхідно описати її за допомогою
      оператора REAL, при цьому застосовувати
      DIMENSION непотрібно.
      WRITE(6,*)'Введіть значення TN,TK,DT'
      READ(5,*) TN,TK,DT
      N=(TK-TN)/DT+1
        DT=(TK-TN)/(N-1)

      A=TN
      DO 3 I=1,N
```

```
I
      MU(I)=ATAN(ABS(ALOG(A)/EXP(2.*A))**(1./3.))
         WRITE(6,*)'При A=',A,'  значення MU=',MU(I)
      3  A=A+DT

         Блок I можливо замінити блоком II
         I=1
         DO 3 A=TN,TK,DT
   II    MU(I)=
ATAN(ABS(ALOG(A)/EXP(2.*A))**(1./3.))
         WRITE(6,*)'При A=',A,'  значення MU=',MU(I)
      3  I=I+1

         PAUSE
         STOP
         END
```

Знаходження парного числа M:

```
         B=M/2.
         K=M/2.
         IF(B.EQ.K) WRITE(6,*)' Число M є парним'
```

Завдання:

В масиві A(200) знайти добуток елементів,що стоять на непарних місцях.

Фрагмент програми:

```
         DIMENSION A(200)
         READ(5,*) A
         P=1.
         DO 2 I=1,200,2
      2  P=P*A(I)
```

Слід звернути увагу, що після закінчення циклу змінна циклу має значення більше кінцевого значення в операторі циклу **DO.**

Завдання:

Знайти максимальне значення элементу масива А.

Фрагмент програми:

```
      DIMENSION A(200)
      READ(5,*)A
      B=A(1)
      DO 3 I=2,200
      IF(A(I).GT.B) B=A(I)   або  B=AMAX1(B,A(I))
    3 CONTINUE
      WRITE(6,*)'Максимальне значення масива
A=',B
```

Завдання:

Знайти мінімальне значення масива А і номер цього элемента.

```
      DIMENSION A(200)
      READ(5,*)A
      NC=1
      C=A(1)
      DO 4 I=2,200
      IF (A(I).GE.C) GO TO 4
      C=A(I)
      NC=I
    4 CONTINUE
      WRITE(6,*)' Мінімальне значення має елемент
під номером ',NC
      WRITE(6,*)'Це значення  дорівнює ',C
```

Завдання:

В одновимірному масиві А(200) поміняти місцями максимальне и мінімальне значення.

Необхідно знайти мінімальний елемент і його номер, потім знайти максимальний елемент і його номер і поміняти ці значення місцями.

```
      REAL A(200)
      WRITE(6,*)'Введіть кількість елементів
масива A
     * та їх значення'
      READ(5,*) N,(A(I),I=1,N)
      AMI=A(1) Задаємо початкові значення для
      нзнаходження максимума й мінімума
      AMA=A(1)
      NMI=1 Задаємо початкові значення для
     визначення номерів циклів, коли знаходиться
     максимум і мінімум.
      NMA=1
      DO 8 I=2,N
C     Блок знаходження максимального елемента і
його  C       номера
      IF(A(I).LE.AMA) GO TO 3
      AMA=A(I)
      NMA=I
C     Блок знаходження мінімального елемента і
його
C     номера
    3 IF(A(I).LT.AMI)  AMI=A(I)
      IF(A(I).EQ.AMI)  NMI=I
      CONTINUE
C     Найдено:
      WRITE(6,*)'Максимальний елемент:
    A(',NMA,') =',AMA
      WRITE(6,*)'Мінімальний елемент: A(',NMI,')
    =',AMI
C     Міняємо місцями:
      A(NMI)=AMA
      A(NMA)=AMI
      PAUSE
      STOP
```

```
      END
```

Сортування масиву

Завдання:

Одномірний масив А (200) відсортувати за спаданням.

У циклі порівнюємо по черзі два елемента масиву, що стоять поруч. І, якщо наступний елемент більше попереднього, то міняємо їх місцями. За один цикл від 1 до N-1 поміняються місцями всі елементи, значення яких задовольняє умові. Якщо ж цикл помістити усередині такого ж циклу, то поміняються місцями всі елементи і масив буде відсортований.

```
C     Перший метод сортування масиву
      REAL A(200)
      WRITE(6,*)'Введіть кількість елементів
масиву А          *та їх значення'
      READ(5,*) N,(A(I),I=1,N)
      DO 8 J=1,N–1
      DO 7 I=1,N–1
      IF(.NOT.(A(I+1).GT.A(I))) GO TO 7 або:
      IF(A(I+1).LE.A(I)) GO TO 7
C     Блок перестановки сусідніх чисел в масиві
      B=A(I)
      A(I)=A(I+1)
      A(I+1)=B
    7 CONTINUE
    8 CONTINUE
      WRITE(6,*)'Відсортований масив:'
      DO 28 I=1,N,2
   28 WRITE(6,*) 'A(',I,')=',A(I),'
A(',I+1,')=',A(I+1)
      PAUSE
      STOP
      END
```

```
C       Другий метод сортування масиву
        REAL A(200)
        WRITE(6,*) 'Введіть кількість елементів
масиву A                *та їх значення'
        READ(5,*) N,(A(I),I=1,N)
        DO 8 J=1,N–1 Задаємо зовнішній цикл
        AMA=A(J)
        DO 7  I=J,N
        IF(A(I).GT.AMA)  AMA=A(I)
    7   IF(A(I).EQ.AMA)  NMA=I
C       Блок перестановки
        B=A(J)
        A(J)=AMA
        A(NMA)=B
    8   CONTINUE
        WRITE(6,*) 'Відсортований масив:'
        DO 28 I=1,N
   28   WRITE(6,*) 'A(',I,')=',A(I)
        PAUSE
        STOP
        END
```

Завдання:

В одновимірному масиві з N елементів виділити і записати в окремі масиви:

1). Елементи, значення яких менше нуля.

2). Елементи, значення яких від 0 до 100.

3). Елементи більше 100.

Вивести, скільки відсотків від загальної кількості складає кількість першого, другого і третього масивів. Скористатися структурою IF THEN ELSE.

```
        DIMENSION
A(1000),A1(1000),A2(1000),A3(1000)
   15   J=1
```

```
      K=1
      L=0
      WRITE(6,*)'Введіть  кількість елементів
      масиву та їх значення'
      READ(5,*) N,(A(I),I=1,N)
      DO 10 I=1,N
C     Блок запису в масив A1 від'ємних елементів
масиву A:
      IF(A(I).LT.0.) THEN
      A1(J)=A(I)
      J=J+1
C     В масив A2 заносимо значення елементів
масиву A від 1 до 100.
      ELSE IF (A(I).GE.0..AND.A(I).LE.100.) THEN
      A2(K)=A(I)
      K=K+1
C     Блок запису в масив A3 елементів масиву A,
більших 100.
      ELSE
      L=L+1
      END IF
      A3(L)=A(I)
      END IF
   10 CONTINUE
C     Обчислення відсотків
      P1=(J-1.)/N*100.
      P2=(K-1.)/N*100.
      або: P2=100.*(K-1)/N
      P3 = L * 100. / N
      WRITE(6,*)'Кількість від'ємних чисел, в
%=',P1
      WRITE(6,*)' Кількість чисел від 0 до 100, в
%=',P2
      WRITE(6,*)' Кількість чисел, які більше
100',P3
      PAUSE
      GO TO 15
      STOP
```

END

Двовимірні масиви

Двовимірні масиви описані в операторі

DIMENSION A(N1,N2)

де A – імя масиву;
N1,N2 – кількість рядків і стовпців відповідно.
Приклад заповнення комірки масиву:**A(2,4)=3.3**

Головна особливість роботи з двовимірними масивами - в пам'яті елементи розташовані по стовпцях.

Завдання:

1) Знайти суму елементів двовимірного масиву A(3,5):

Фрагмент програми:

```
DIMENSION A(3,5)
          READ(5,*)A
        S=0.
        DO 18 J=1,5
        DO 8 I=1,3
    7   S=S+A(I,J)
   18   CONTINUE
```

2) Знайти суму всіх додатніх елементів

Фрагмент програми:

```
          DIMENSION A(3,5)
    …
          S=0.
          DO  18  I=1,3
          DO   8 J=1,5
```

```
      8  IF(A(I,J).GT.0.)  S=S+A(I,J)
     18  CONTINUE
```

3) Знайти максимальне значення масиву

Фрагмент програми:

```
          AM=A(1,1)
          DO 18 J=1,5
          DO  8  I=1,3
      8   AM=AMAX1(AM,A(I,J))
```
Або: IF(A(I,J).GT.AM) AM=A(I,J)
```
     18  CONTINUE
```

4) Знайти мінімальний елемент, номер рядка і номер стовпця, в яких він знаходиться

Фрагмент програми:

```
          AMI=A(1,1)
          DO 8 J=1,5
          DO 8 I=1,3
          IF (A(I,J).GT.AMI) GO TO 8
          AMI=A(I,J)
          NSTR=I
          NSTB=J
     8    CONTINUE
```

5) Транспонувати матрицю

Фрагмент програми:

```
DIMENSION A(3,5),B(5,3)
          DO 11 L=1,5
          DO 11 K=1,3
10        B(L,K)=A(K,L)
```

6) Знайти суму елементів головної діагоналі

Фрагмент програми:

```
      S=0.
      DO 16 I=1,3
16  S=S+A(I,I)
```

7) Знайти добуток всіх елементів,за виключенням тих, що стоять під неголовною діагоналлю і на діагоналі

Фрагмент програми:

```
      PR=1.
      DO 12 I=1,3
      Цикл, в якому змінюється номер
      рядка.
      D0 12 J=1,5-I
      Цикл, в якому змінюється номер
      стовпця  (якщо в добуток залучити неголовну
      діагональ, тоді  J=1,6–I).
   12 PR=PR*A(I,J)
```

+	+	+	+	
+	+	+		
+	+			

8) Знайти кількість додатних елементів кожного стовпця. Всі ці значення занести в окремий масив.

Фрагмент програми:

```
REAL A(3,5), B(5)
      READ(5,*) A
C     Зовнішній цикл
      DO 15 N=1,5
C     Внутрішній цикл
      B(N)=0.
      DO 16 M=1,3
16     IF(A(M,N).GT.0.) B(N)=B(N)+1
```

```
15      CONTINUE
```

9) Поміняти місцями останній і перший стовпці масива.

Фрагмент програми:

```
      REAL A(3,5)
      READ(5,*) A
      DO 17 I=1,3
      C=A(I,1)
      A(I,1)=A(I,5)
17    A(I,5)=C
```

10) Знайти суму елементів кожного рядка і записати їх в масив

Фрагмент програми:

```
      REAL A(3,5), C(3)
      READ (5,*)A
      DO 10 I=1,3
      C(I)=0.
      DO 9 J=1,5
    7 C(I)=C(I)+A(I,J)
    8 CONTINUE
```

5. Чисельні методи в Фортрані, що застосовуються при розв'язанні задач механіки твердого деформівного тіла.

Для успішного розв'язання задач механіки деформівного твердого тіла необхідно володіти цілим рядом числених методів. Зокрема, необхідно володіти методами розв'язання трансцендентних рівнянь, методами інтерполювання функцій, розв'язання систем лінійних алгебраїчних рівнянь, тощо. Ці методи є невід'ємними складовими загальної схеми розв'язання задач механіки деформівного твердого тіла, які використовуються у чисельних алгоритмах розв'язання таких задач.

Традиційно мова програмування Фортран вигідно вирізнялася серед інших мов програмування наявністю великої бібліотеки чисельних алгоритмів, зокрема чисельних методів розв'язання задач механіки. Тому при виборі програмних засобів реалізації того чи іншого алгоритму розв'язання механічних задач доцільно скористатися готовими схемами чисельного розрахунку, реалізованими на мові Фортран. Нижче розглянемо основні чисельні методи, що використовуються в МТДТ, та особливості їх реалізації на мові Фортран.

5.1. Метод Ньютона розв'язання трансцендентних рівнянь.

Метод Ньютона (також відомий як метод дотичних) — це ітераційний чисельний метод знаходження кореня заданої нелінійної функції. Метод був уперше запропонований англійським фізиком, математиком і астрономом Ісааком Ньютоном (1643-1727). Пошук розв'язку здійснюється шляхом побудови послідовних наближень і заснований на принципах простої ітерації. Метод має квадратичну збіжність.

Алгоритм розв'язання задач за допомогою методу Ньютона

1) визначаємо інтервал (якщо він не заданий), якому буде належати корінь рівняння. Звуження інтервалу можна робити методом половинного ділення.

2) знаходимо $f(x)$ і $f''(x)$, причому $f'(x) \neq 0$ при $x \in [a,b]$, $f'(x)$ і $f''(x)$ повинні зберігати знак на відрізку $[a,b]$

3) вибираємо один з кінців відрізка $[a,b]$ за x_0, виходячи з того, що повинне виконуватися наступну умову $f(x_0) f''(x_0) > 0$.

4) обчислюємо $x_i = x_{i-1} - \dfrac{f(x_{i-1})}{f'(x_{i-1})}$, поки не будуть збігатися десяткові знаки, які необхідні у відповіді, або задана точність ε - до виконання нерівності $|x_i - x_{i-1}| < \varepsilon$.

Приклади розв'язання рівнянь за допомогою методу Ньютона

Розглянемо застосування методу Ньютона на прикладах.

1) Нехай задана функція $f(x) = sin 2x - ln x$, якщо $1{,}3 < x < 1{,}5$. Необхідно знайти корінь рівняння з точністю до 0,0001.

Знайдемо першу й другу похідні вихідної функції:

$$f'_x = 2cos 2x - \frac{1}{x}$$

$$f''_x = -4 sin 2x + \frac{1}{x^2}$$

x	$f(x)$	$f''(x)$
1,3	0,253137	-1,47029
1,5	-0,26435	-0,12004

Оскільки $f(x)f''(x)>0$ при $x=1{,}5$, то за x_0, беремо $x=1{,}5$.

Далі, $x_i = x_{i-1} - \dfrac{f(x_{i-1})}{f'(x_{i-1})}$.

x	$f(x)$	$f'(x)$	$x_n - x_{n-1}$
1,5	-0,2643451	-2,64665166	
1,40012093	-0,001798363	-2,59883069	-0,0998707
1,39942894	-0,0000001988	-2,59825545	-0,00059199
1,39942887	-3*10-17	-2,59825539	-0,00000007
1,39942887			

Оскільки $|x_3 - x_2| < 0{,}0001$, то на даному кроці можна зупинитися.

Відповідь: $x \approx 1{,}39943$.

2) Розв'язати рівняння $f(x) = e^{2x} + 3x - 4$ з точністю $\varepsilon = 10^{-3}$.

Так як невідомий інтервал, якому належить корінь рівняння, то для локалізації кореня застосуємо графічний спосіб. Перетворимо вихідне рівняння до наступного еквівалентного виду:

$e^{2x} = 4 - 3x$

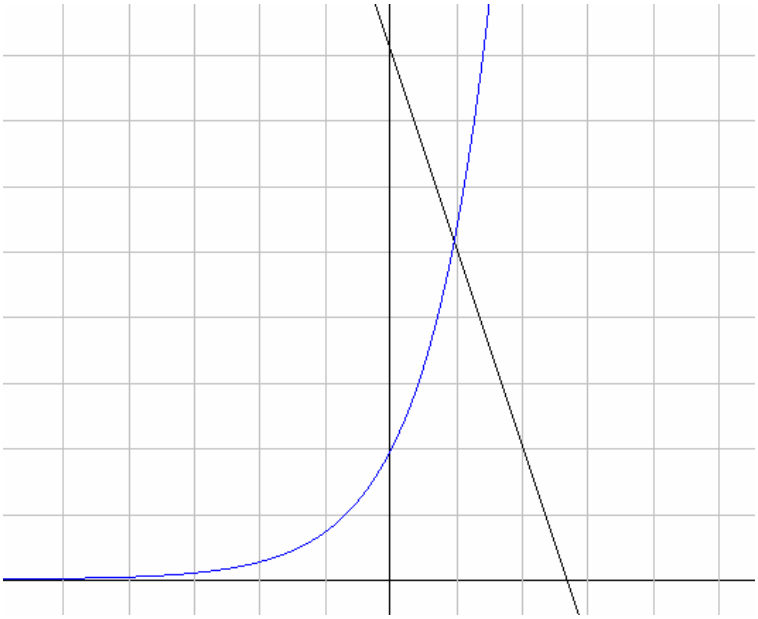


Рис. 3 Графічна інтерпретація вихідного рівняння.

Побудувавши графіки функцій $f_1(x)=e^{2x}$ і $f_2(x)=4-3x$, визначаємо, що в розв'язуваного рівняння є тільки один корінь, що лежить в інтервалі $0{,}4<x<0{,}6$. На даному інтервалі дійсно міститься корінь рівняння, так як $f(0{,}4)\cdot f(0{,}6)<0$.

Уточнимо значення кореня з необхідною точністю, користуючись методом Ньютона.

Для коректного використання даного методу необхідно визначити поводження першої й другої похідної функції на інтервалі уточнення кореня й правильно вибрати початкове наближення x_0.

Для функції $f(x)$ маємо:

$f'(x)=2e^{2x}+3$ і $f''(x)=4e^{2x}$ - додатні у всій області визначення функції. За початкове наближення необхідно вибрати праву границю інтервалу $x_0=0{,}4$, для якої виконується нерівність: $f(0{,}6)\cdot f''(0{,}6)>0$.

Результати обчислень:

Номер ітерації	x0	F(x0)	F'(x0)	R	xn-xn-1

0	0,6	1,107982086	9,615964	0,115223192	
1	0,484777	0,083308362	8,257956	0,010088255	-0,115223
2	0,474689	0,000690164	8,153249	0,000084649	-0,010088
3	0,474604	0,000001368	8,152379	0,000000168	-0,000085

Оскільки на третьому кроці $|x_3 - x_2| < \varepsilon$, то подальші ітерації можна не виконувати.

Відповідь: $x \approx 0{,}474604$.

Приклад програми обчислення коренів трансцендентного рівняння методом Ньютона.

```
PROGRAM Newton
c Solving equation f(x)=0 by Newton method.
REAL a,b,eps
REAL x0,x1,dx
OPEN(unit=1,name='data.txt',status='old')
OPEN(unit=2,name='res.txt',status='new')
c a=-1.5
c b=1.5
c eps=0.001
c dx=0.001
READ(1,3)a
READ(1,3)b
READ(1,3)eps
READ(1,3)dx
3 FORMAT(F5.3)
x0=a
x1=x0-f(x0)*dx/(f(x0+dx)-f(x0))
1 if ((abs(x0-x1)).LE.eps) goto 2
x0=x1
x1=x0-f(x0)*dx/(f(x0+dx)-f(x0))
GOTO 1
2 CONTINUE
WRITE(2,4)x0
```

```
4 FORMAT(F5.3)
CLOSE(2)
CLOSE(1)
STOP
END
```

Підпрограма задання рівняння

```
f.for
function f(x)
f=SIN(x)+x-1
return
end
```

файл вхідних даних

```
Data.txt
-1.5 1.5 0.001 0.001
```

Файл результатів

```
res.txt
0.511
```

Завдання.

1) Скласти программу розв’язання рівняння $f(x) = 3x - 18 ln\, x$ з точністю $\varepsilon = 10^{-3}$ за допомогою метода Ньютона.

2) Скласти программу розв’язати рівняння $f(x) = x^4 - 18x - 10$ з точністю $\varepsilon = 10^{-3}$.

3) Скласти программу розв’язати рівняння $f(x) = x^3 - 6x^2 + 3x + 11$ з точністю $\varepsilon = 10^{-3}$.

4) Скласти программу розв'язати рівняння $f(x)=x^3+4\sin x-1$ з точністю $\varepsilon=10^{-3}$.

5) Скласти программу розв'язати рівняння $f(x)=x^3+2\sin x-3$ з точністю $\varepsilon=10^{-3}$.

5.2. Інтерполяція функцій

При розв'язанні більшості обчислювальних задач доводиться мати справу з функціями, заданими таблично, а не аналітично. У цьому випадку додаткові питання виникають навіть тоді, коли треба визначити значення функції в певній точці. Як правило, ця задача носить допоміжний характер, але зараз ми її розглянемо як самостійну.

Розглянемо постановку загальної задачі інтерполяції. Нехай $f(x)$ – визначена на $[a,b]$ і належить деякому класу $C^k[a,b]$.

Задана сітка вузлів $a \le x_0 < x_1 < \ldots < x_n \le b$.

Необхідно побудувати функцію

$$\varphi(x)=\sum_{k=0}^{n} a_k \varphi_k(x),$$

лінійну відносно функцій $\varphi_k(x)$ і таку, що виконується умова

$$\varphi(x_i)=\sum_{k} a_k \varphi_k(x_i)=y_i,$$

причому, система $\{\varphi_k(x)\}_{k=0,\ldots,n}$ — лінійно незалежна.

Вибір системи $\{\varphi_k(x)\}$ визначається класом функцій $f(x)$. Частинний випадок — інтерполяція многочленами $\{\varphi_k(x)\}=\{x^k\}, k=0,1,\ldots,n$.

Лінійна інтерполяція.

При лінійній інтерполяції будується ламана, яка проходить через точки (X_i, Y_i), $i = 0,1,2,...,n$, тобто співпадаюча із шуканою функцією у вузлах інтерполяції і лінійна на кожній ділянці (X_i, X_{i+1}) при $i = 0,1,2,...,n-1$.

При $X_i \le X \le X_{i+1}$ значення побудованої функції $\varphi(X)$ будуть обчислюватися по формулі

$$\varphi(X) = Y_i + (X - X_i)\frac{(Y_{i+1} - Y_i)}{(X_{i+1} - X_i)}.$$

Інтерполяційний поліном Лагранжа.

Інший варіант інтерполяції - шукати функцію у вигляді многочлена ступеня n:

$$\varphi(X) = P_n(X) = C_n X^n + C_{n-1} X^{n-1} + ... + C_1 X + C_0.$$

Умови збігу значень інтерполюючої функції у точках X_i з величинами Y_i приймуть вигляд системи

$$\begin{cases} C_0 + C_1 X_1 + ... + C_{n-1} X_1^{n-1} + C_n X_1^n = Y_1 \\ C_0 + C_1 X_2 + ... + C_{n-1} X_2^{n-1} + C_n X_2^n = Y_2 \\ \qquad\qquad ..... \\ C_0 + C_1 X_n + ... + C_{n-1} X_n^{n-1} + C_n X_n^n = Y_n \end{cases}$$

$(n+1)$-го лінійного рівняння з $n+1$ невідомим.

Цей поліном запишеться у вигляді:

$$P_n(x) = \sum_{i=0}^{n} Y_i \prod_{\substack{j=0 \\ j \ne i}}^{n} \frac{X - X_j}{X_i - X_j},$$

де $P_n(X_j) = Y_j$ при j=0,1,2,...,n.

Інтерполяція сплайнами.

Інтерполяція многочленом Лагранжа або Ньютона на відрізку $[a,b]$ з використанням великої кількості вузлів

інтерполяції часто призводить до поганого наближення, тобто відбувається значне накопичення похибок в процесі обчислень. Крім того, внаслідок розбіжності процес інтерполяції із збільшенням числа вузлів не обов'язково приводить до підвищення точності. Для того, щоб уникнути значних похибок, весь відрізок $[a,b]$ розбивають на часткові відрізки й на кожному із часткових відрізків наближено заміняють функцію $f(x)$ многочленом невисокого ступеня (так звана кусково-поліноміальна інтерполяція). Одним із способів інтерполяції на всьому відрізку є інтерполяція за допомогою сплайн-функцій. Сплайн-функцією або сплайном називають кусочно-поліноміальну функцію, визначену на відрізку $[a,b]$, що має на цьому відрізку деяке число неперервних похідних.

Слово "сплайн" (англійське spline) означає гнучку лінійку, використовувану для проведення гладких кривих через задані точки площини.

Перевага сплайнів перед звичайною інтерполяцією є, по-перше, їхня збіжність, і, по-друге, стійкість процесу обчислень.

Розглянемо частинний, але поширений в обчислювальній практиці випадок, коли сплайн визначається за допомогою многочленів третього ступеня (кубічний сплайн).

Нехай на $[a,b]$ задана неперервна функція $f(x)$. Введемо вузли (сітку):

$$a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b,$$

і позначимо $f_i = f(x_i), i \equiv \overline{0,n}.$

Інтерполяційним кубічним сплайном, що відповідає даній функції $f(x)$ і даним вузлам, называється функція $s(x)$, яка задовольняє наступним умовам:

а) на кождом відрізку $[x_{i-1}, x_i]$, $i = \overline{1,n}$ функція $s(x)$ є многочленом третього ступеня;

б) функція $s(x)$, а так само її перша й друга похідні неперервні на $[a,b]$;

в) $s(x_i) = f(x_i), i = \overline{0,n}.$

Остання умова називається умовою інтерполяції.

На кожному з відрізків $[x_{i-1}, x_i], i = \overline{1,n}$ будемо шукати функцію $s(x) = s_i(x)$ у вигляді многочлена третього ступеня

$$s_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3,$$

$$x_{i-1} \le x \le x_i, \quad i = \overline{1,n},$$

$$a_i = f(x_i), \quad i = \overline{1,n}.$$

Довизначимо, крім того, $a_0 = f(x_0)$.

для визначення коефіцієнтів c_i одержуємо систему рівнянь

$$h_i c_{i-1} + 2(h_i + h_{i+1})c_i + h_{i+1}c_{i+1} =$$

$$= 6\left(\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}\right), c_0 = c_n = 0,$$

де $h_i = x_i - x_{i-1}, i = \overline{1,n-1}$

В силу діагональної переваги ця система має єдиний розв’язок. Оскільки матриця системи - трьохдіагона, то розв’язок можна знайти методом прогону. По знайдених коефіцієнтах c_i коефіцієнти b_i і d_i визначаються з допомогою формул

$$d_i = \frac{c_i - c_{i-1}}{h_i}, b_i = \frac{h_i}{2} c_i - \frac{h_i^2}{6} d_i + \frac{f_i - f_{i-1}}{h_i}, i = \overline{1,n}.$$

Приклади інтерполяції функцій за допомогою поліному Лагранжа

Розглянемо інтерполяцію функцій за допомогою інтерполяційного поліному Лагранжа на прикладах.

```
PROGRAM InterpLagrang
REAL x(100),y(100),x1(100)
REAL p,x_,L_
INTEGER i,j,k,N,M
OPEN(unit=1,file='lagr.dat',status='old')
OPEN(unit=2,file='lagr.res',status='new')
READ(1,*)N
IF(N.GE.100) goto 1
READ(1,*)(x(i),i=1,N)
READ(1,*)(y(i),i=1,N)
READ(1,*)M
IF(M.GE.100) GOTO 1
READ(1,*)(x1(i),i=1,M)
WRITE(2,*)M
WRITE (2,*)
DO 2 i=1,M
x_=x1(i)
L_=0
DO 3 j=1,N
p=1
DO 4 k=1,j-1
p=p*(x_-x(k))/(x(j)-x(k))
4          CONTINUE
DO 5 k=j+1,N
p=p*(x_-x(k))/(x(j)-x(k))
5          CONTINUE
L_=L_+y(j)*p
3      CONTINUE
WTITE(2,*)L_
2   CONTINUE
```

```
1   CLOSE(2)
CLOSE(1)
STOP
END
```

Структура файлу із вхідними даними.

```
N
x(1) x(2) ... x(N)
y(1) y(2) ... y(N)
M
x_(1) x_(2) ... x_(M)
```

N - кількість точок, вузлів інтерполяції.

{x(i),y(i)|i=1,N} - координати точок вузлів інтерполяції.

M - кількість точок, в яких треба визначити значення інтерполяційного поліному Лагранжа.

{x_(i)|i=1,M} - x координати цих точок.

Тестовый варіант файлу із вхідними даними.

Lagr.dat

```
4
1 3 5 7
1 9 25 49
6
0 1 2 3 4 5
```

Результат работи програми.

Lagr.res

```
6
0.000000
1.000000
4.000000
9.000000
16.00000
```

25.00000

Завдання.

1. Протестувати програму лінійної інтерполяції за допомогою полінома Лагранжа для наступних даних:

X	0	1	2	0.5
y	0	1	2	?

Знайти похибку обчислень.

2. Протестувати програму лінійної інтерполяції за допомогою полінома Лагранжа для наступних даних:

X	0	1	2	3	4	5	6	7	2.7
y	3	3	3	3	3	3	3	3	?

Знайти похибку обчислень.

5.3. Задачі лінійної алгебри

Метод Гауса розв'язання систем лінійних алгебраїчних рівнянь.

Запишемо систему $Ax = f$, у розгорнутому виді

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m = f_1, \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2m}x_m = f_2, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mm}x_m = f_m. \end{aligned}$$

Метод Гаусса полягає в послідовному виключенні невідомих із цієї системи. Припустимо, що $a_{11} \neq 0$. Послідовно множачи перше рівняння на $-\frac{a_{i1}}{a_{11}}$ й складаючи з i-м рівняння, виключимо x_1 із всіх рівнянь крім першого. Одержимо систему

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1_m}x_m = f_1,$$
$$a_{22}^{(1)}x_2 + ... + a_{2m}^{(1)}x_m = f_2^{(1)},$$
$$.....$$
$$a_{m2}^{(1)}x_2 + ... + a_{mm}^{(1)}x_m = f_m^{(1)},$$

де $a_{ij}^{(l)} = a_{ij} - \frac{a_{il}a_{lj}}{a_{ll}}$, $f_i^{(l)} = f_i - \frac{a_{il}f_l}{a_{ll}}$, $i, j = 2,3...m.$

Аналогічним образом з отриманої системи виключимо x_2. Послідовно, крім всі невідомого, одержимо систему трикутного виду

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1k}x_k + ... + a_{1m}x_m = f_1,$$
$$a_{22}^{(1)}x_2 + ... + a_{2k}^{(1)}x_k + ... + a_{2m}^{(1)}x_m = f_2^{(1)},$$
$$.....$$
$$a_{m-1,m-1}^{(m-1)}x_{m-1} + a_{m-1,m}^{(m-1)}x_m = f_{m-1}^{(m-1)},$$
$$a_{m,m}^{(m-1)}x_m = f_m^{(m-1)}.$$

Описана процедура називається прямим ходом методу Гаусса. Помітимо, що її виконання було можливо за умови, що всі $a_{i,i}^{(l)}$, $l = 1,2...m-1$ не дорівнюють нулю.

Виконуючи послідовні підстановки в останній системі, (починаючи з останнього рівняння) можна одержати всі значення невідомих.

$$x_m = \frac{f_m^{(m-1)}}{a_{m,m}^{(m-1)}},\ x_i = \frac{1}{a_{i,i}^{(i-1)}}(f_i^{(i-1)} - \sum_{j=i-1}^{m} a_{ij}^{(i-1)}x_j).$$

Ця процедура одержала назву зворотний хід методу Гауса.

Приклад програми розв'язання систем лінійних алгебраїчних рівнянь за методом Гауса

```
PROGRAM GAUSSMETHOD
IMPLICIT NONE
INTEGER,PARAMETER :: N=3,N1=N+1
INTEGER :: I,J
REAL*8    :: A(N,N1)
REAL*8    :: V(N)
OPEN(10,FILE="FILE.TXT")
! ВВОД МАТРИЦІ A(N,N+1). ОСТАННІЙ СТОВПЧИК ---
ВЕКТОР ПРАВИХ ЧАСТИН СИСТЕМИ РІВНЯНЬ Ax=b
A( 1, 1)= 1.0D+00
A( 1, 2)=-3.0D+00
A( 1, 3)= 2.0D+00
A( 1, 4)= 7.0D+00
A( 2, 1)= 4.0D+00
A( 2, 2)= 6.0D+00
A( 2, 3)= 1.0D+00
A( 2, 4)= 3.0D+00
A( 3, 1)= 2.0D+00
A( 3, 2)= 1.0D+00
A( 3, 3)=-2.0D+00
A( 3, 4)= 1.0D+00
! ВВОД МАТРИЦІ A(N,N+1). ОСТАННІЙ СТОВПЧИК ---
ВЕКТОР ПРАВИХ ЧАСТИН СИСТЕМИ РІВНЯНЬ Ax=b

WRITE(*,*)  "MATRIX A"
WRITE(10,*) "MATRIX A"
DO I=1,N
WRITE(*,*)  (A(I,J),J=1,N+1)
WRITE(10,*) (A(I,J),J=1,N+1)
END DO

CALL GAUSS(N,A,V)

! РОЗВ'ЯЗАННЯ СИСТЕМИ РІВНЯНЬ
WRITE(*,*)  "SOLUTION"
WRITE(10,*) "SOLUTION"
WRITE(*,*)  (V(I),I=1,N)
WRITE(10,*) (V(I),I=1,N)
```

```
! РОЗВ'ЯЗАННЯ СИСТЕМИ РІВНЯНЬ
WRITE(*,*)  "END"
WRITE(10,*) "END"
CLOSE(10)
END

SUBROUTINE GAUSS(N,A,V)
IMPLICIT NONE

INTEGER,INTENT(IN) :: N
REAL*8,INTENT(INOUT) :: A(N,N+1)
REAL*8,INTENT(OUT)   :: V(N)

INTEGER          :: I,J,K,IMAX,JMAX
REAL*8           :: MAX,S,W(N)

INTEGER          :: INDEX(N),M

DO I=1,N
V(I)=0.0D+00
INDEX(I)=I
END DO

DO K=1,N-1
MAX=0.0D+00
DO I=K,N
DO J=K,N
IF (MAX.LT.ABS(A(I,J))) THEN
MAX=ABS(A(I,J))
IMAX=I
JMAX=J
END IF
END DO
END DO

IF (MAX.EQ.0.0D+00) THEN
WRITE(*,*) "det(A)=0"
WRITE(*,*) "STOP"
```

```
STOP
END IF

DO I=1,N
S=A(I,K)
A(I,K)=A(I,JMAX)
A(I,JMAX)=S

        M=INDEX(K)
        INDEX(K)=INDEX(JMAX)
        INDEX(JMAX)=M
      END DO

DO J=1,N+1
S=A(K,J)
A(K,J)=A(IMAX,J)
A(IMAX,J)=S
END DO

S=A(K,K)
DO J=K,N+1
A(K,J)=A(K,J)/S
END DO

DO I=K+1,N
S=A(I,K)
DO J=K,N+1
A(I,J)=A(I,J)-S*A(K,J)
END DO
END DO
END DO

IF (A(N,N).EQ.0.0D+00) THEN
WRITE(*,*) "det(A)=0"
WRITE(*,*) "STOP"
STOP
END IF
```

```
S=A(N,N)
DO J=N,N+1
A(N,J)=A(N,J)/S
END DO

W(N)=A(N,N+1)
DO I=N-1,1,-1
S=0.0D+00
DO J=I+1,N
S=S+A(I,J)*W(J)
END DO
W(I)=A(I,N+1)-S
END DO

DO I=1,N
V(INDEX(I))=W(I)
END DO

RETURN
END
```

Завдання.

1) Розв’язати методом Гауса систему рівнянь

$$\begin{cases} x_1 + x_2 + x_3 = 6 \\ 2x_1 - 3x_2 + 4x_3 = 4 \\ x_1 - 5x_2 + 6x_3 = -1 \end{cases}$$

2) Розв’язати методом Гауса систему рівнянь

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

3) Розв’язати методом Гауса систему рівнянь

$$\begin{cases} 2x_1 + 3x_2 + x_3 = 6 \\ x_1 - 2x_2 + 3x_3 = 2 \\ x_1 - 5x_2 + 5x_3 = 1. \end{cases}$$

5.4. Числені методи розв'язання задачі Коші.

Задача для звичайного диференційного рівняння першого порядку для функції однієй змінної ставиться наступним чином

$$\begin{cases} \dfrac{du}{dx} = u'(x) = f(x,u) \\ u(x_0) = u_0 \end{cases}$$

Більш загальна постановка задачі Коші для диференційного рівняння n-го порядку

$$\begin{cases} u^{(n)}(x) = f(x,u,u',...,u^{(n-1)}) \\ u(x_0) = u_0 \\ u'(x_0) = u'_0 \\ \dots \\ u^{(n-1)}(x_0) = u_0^{(n-1)} \end{cases}$$

Тут $u_0, u_0', ..., u_0^{(n-1)}$ - задані числа (початкові умови).

Ця задача за допомогою заміни змінних

$u^{(k)}(x) = z_k(x), k = 0,1,...,n$,

$u^0(x) = u(x) = z_0(x)$.

зводиться до системи диференціальних рівнянь першого порядку:

$$\begin{cases} z'_{n-1} = f(x, z_0, z_1, ..., z_{n-1}) \\ z'_{n-2} = (u^{(n-2)})' = u^{(n-1)} = z_{n-1} \\ \dots \\ z'_0 = z_1 \\ z_i(x_0) = u_0^{(i)}, \quad i = 0,1,...,n-1 \end{cases}$$

Методи Рунге-Кутта.

Методи Рунге-Кутта — це група однокрокових ітераційних методів розв'язання задачі Коші. При переході з точки (x_k, y_k) в точку $(x_{k+1}, y_{k+!})$ використовується лише інформація про попередню точку (x_k, y_k). Цій умові відповідає такий загальний запис ітераційної процедури

$$y_{k+1} = y_k + h\kappa(x_k, y_k),$$

де $\kappa(x_k, y_k)$ виражається через значення функції $f(x, y)$ в точці (x_k, y_k) чи близьким до неї (зсунутим на долю кроку).

Метод Рунге-Кутта другого порядку носить назву "предиктор-коректор" та має вигляд:

$$\begin{cases} \overline{y_{k+1}} = y_k + hf(x_k, y_k) \\ y_{k+1} = y_k + h\dfrac{f(x_k, y_k) + f(x_{k+1}, \overline{y_{k+1}})}{2} \end{cases}$$

На першому етапі "передбачаємо" значення $\overline{y_{k+1}}$ по методу Ейлера. На другому етапі це значення коректується шляхом осереднення кутових коефіцієнтів в точках (x_k, y_k) і $(x_{k+1}, \overline{y_{k+1}})$. За рахунок корекції, точність даного методу і підвищується на порядок в порівнянні з методом Ейлера.

Метод Рунге-Кутта четвертого порядку точності (найбільш уживаний на практиці) сформульований наступним чином

$$\begin{cases} y_{k+!} = y_k + \dfrac{h}{6}(\kappa_1 + 2\kappa_2 + 2\kappa_3 + \kappa_4), \\ y_0 = u_0 \end{cases}$$

де

$$\kappa_1 = f(x_k, y_k),$$

$$\kappa_2 = f(x_k + \frac{h}{2}, y_k + \frac{\kappa_1}{2}),$$

$$\kappa_3 = f(x_k + \frac{h}{2}, y_k + \frac{\kappa_2}{2}),$$

$$\kappa_4 = f(x_k + h, y_k + \kappa_3)$$

Якщо $f(x,y) \in C^4(\overline{G})$, то похибка процедури $\approx \underline{\underline{O}}(h^4)$.

При практичному застосуванні методів Рунге-Кутта виникає питання: яка з формул більш придатна для практичної реалізації. Якщо відомо, що $f(x,y)$ - достатньо гладка функція, наприклад, $f \in C^4\left(\overline{G}\right)$, то найбільш ефективна процедура методу 4-го порядку. Якщо ж гладкість функції $f(x,y)$ недостатня, то краще використовувати методи методи другого та третього порядку.

Приклад програми розв'язання задачі Коші за допомогою методу Рунге-Кутта 2-го порядку (метод предиктор-коректор)

```
MODULE RUNGE2
IMPLICIT NONE
```

```
CONTAINS

      REAL(8) FUNCTION F1(X,Y)
            IMPLICIT NONE
            REAL(8) X,Y
            F1=(Y-X*Y-X*X)/X
      END FUNCTION

      REAL(8) FUNCTION FP1(X)
            IMPLICIT NONE
            REAL(8) X
            FP1=X*(2.0D0*EXP(1-X)-1)
      END FUNCTION

      REAL(8) FUNCTION F2(X,Y)
            IMPLICIT NONE
            REAL(8) X,Y
            F2=(2.0D0*X*Y+3.0D0*Y)/X**2
      END FUNCTION

      REAL(8) FUNCTION FP2(X)
            IMPLICIT NONE
            REAL(8) X
            FP2=X*X*EXP(3.0D0-3.0D0/X)
      END FUNCTION

      REAL(8) FUNCTION F3(X,Y)
            IMPLICIT NONE
            REAL(8) X,Y
            F3=(EXP(X-Y)-1.0D0)/X
      END FUNCTION

      REAL(8) FUNCTION FP3(X)
            IMPLICIT NONE
            REAL(8) X
            FP3=LOG(EXP(X)/X)
      END FUNCTION
```

```
END MODULE

PROGRAM METODRUNGE2
USE RUNGE2
IMPLICIT NONE
INTEGER(4) I,LIM,N
PARAMETER (LIM=1000000)
REAL(8) M1(LIM),M2(LIM),M3(LIM)
REAL(8) X0,B,Y0,AF,Y1,MM1,MM2,MM3,HH,HHH,A1,A2
WRITE(*,*) ' THE INITIAL GIVEN PROBLEMS OF KOSHI'
READ(*,*) X0,Y0
WRITE(*,*) ' ENTER X AND NUMBER OF INTERVALS(N)'
READ(*,*) B,N
WRITE(*,*) 'ENTER ALFA'
READ(*,*) AF

!FIRST FUNCTION
HH=(B-X0)/N
Y1=Y0+HH*((1.0D0-
AF)*F1(X0,Y0)+AF*F1(X0+HH/(2.0D0*AF),Y0+HH/(2.0D0*
AF)*F1(X0,Y0)))
DO I=1,N
      HHH=X0+I*HH
      A1=(1-AF)*F1(HHH,Y1)
      A2=AF*F1(HHH+HH/(2*AF),Y1+HH/(2*AF)*F1(HHH,Y
1))
      Y1=Y1+HH*(A1+A2)
      M1(I)=ABS(Y1-FP1(HHH+HH))
END DO
MM1=MAXVAL(M1)
WRITE(*,*) 'METOD RUNGE-KYTTA 2 ORDER'
WRITE(*,'(A20,D15.5)') 'Y(X)=' ,Y1
WRITE(*,'(A20,D15.5)') ' Y(X)(TABL)=' ,FP1(B)
WRITE(*,'(A20,D15.5/)') ' ERR=' ,MM1

!SECOND FUNCTION
```

```
Y1=Y0+HH*((1.0D0-
AF)*F2(X0,Y0)+AF*F2(X0+HH/(2.0D0*AF),Y0+HH/(2.0D0*
AF)*F2(X0,Y0)))
DO I=1,N
      HHH=X0+I*HH
      A1=(1-AF)*F2(HHH,Y1)
      A2=AF*F2(HHH+HH/(2*AF),Y1+HH/(2*AF)*F2(HHH,Y
1))
      Y1=Y1+HH*(A1+A2)
      M2(I)=ABS(Y1-FP2(HHH+HH))
END DO
MM2=MAXVAL(M2)
WRITE(*,*) 'METOD RUNGE-KYTTA 2 ORDER'
WRITE(*,'(A20,D15.5)') 'Y(X)=' ,Y1
WRITE(*,'(A20,D15.5)') ' Y(X)(TABL)=' ,FP2(B)
WRITE(*,'(A20,D15.5/)') ' ERR=' ,MM2

!THIRD FUNCTION
Y1=Y0+HH*((1.0D0-
AF)*F3(X0,Y0)+AF*F3(X0+HH/(2.0D0*AF),Y0+HH/(2.0D0*
AF)*F3(X0,Y0)))
DO I=1,N
      HHH=X0+I*HH
      A1=(1-AF)*F3(HHH,Y1)
      A2=AF*F3(HHH+HH/(2*AF),Y1+HH/(2*AF)*F3(HHH,Y
1))
      Y1=Y1+HH*(A1+A2)
      M3(I)=ABS(Y1-FP3(HHH+HH))
END DO
MM3=MAXVAL(M3)
WRITE(*,*) 'METOD RUNGE-KYTTA 2 ORDER'
WRITE(*,'(A20,D15.5)') 'Y(X)=' ,Y1
WRITE(*,'(A20,D15.5)') 'Y(X)(TABL)=' ,FP3(B)
WRITE(*,'(A20,D15.5/)') 'ERR=' ,MM3

PAUSE
END PROGRAM
```

Приклад програми розв'язання задачі Коші за допомогою методу Рунге-Кутти 4-го порядку

```
IMPLICIT NONE
INTEGER :: I,J,N0
REAL*8  :: ALPHA,BETA,GAMMA,H,T,TMAX
REAL*8, Allocatable :: W(:)
N0=3
Allocate( W(N0) )
 OPEN(100,FILE="SOLUTION.TXT")
! ========== ВХІДНІ ПАРАМЕТРИ ==========
      ALPHA=28.0D+00
      BETA=8.0D+00/3.0D+00
      GAMMA=10.0D+00
! ========== ВХІДНІ ПАРАМЕТРИ ==========
!  TMAX - ПРАВА ГРАНИЦЯ ІНТЕРВАЛУ
ІНТЕГРУВАННЯ
      TMAX=100.0D+00
!  H - КРОК ІНТЕГРУВАННЯ
      H=0.001D+00

      DO I=1,N0
        W(I)=0.0D+00
      END DO
      W(1)=1.0D+00

      T=0.0D+00
      WRITE(100,*) "    T          X(T)       Y(T)       Z(T)"

100   CONTINUE
      IF (T.LE.TMAX) THEN
        CALL RUNGE(N0,ALPHA,BETA,GAMMA,T,H,W)
        T=T+H
        WRITE(*,1000)    T,(W(I),I=1,N0)
        WRITE(100,1000)  T,(W(I),I=1,N0)
        GOTO 100
      END IF
```

```
1000  FORMAT(1000F12.5)
      CLOSE(100)
      END

      SUBROUTINE
PROBLEM(N0,ALPHA,BETA,GAMMA,T,W,G)

      INTEGER, INTENT(IN) :: N0
      REAL*8,INTENT(IN)   :: ALPHA,BETA,GAMMA
      REAL*8,INTENT(IN)   :: T
      REAL*8,INTENT(IN)   :: W(N0)
      REAL*8,INTENT(OUT)  :: G(N0)

      INTEGER             :: I
      DO I=1,N0
        G(I)=0.0D+00
      END DO

      G(1)=-GAMMA*W(1)+GAMMA*W(2)
      G(2)=-W(1)*W(3)+ALPHA*W(1)-W(2)
      G(3)=W(1)*W(2)-BETA*W(3)

      END SUBROUTINE PROBLEM

      SUBROUTINE
RUNGE(N0,ALPHA,BETA,GAMMA,T,H,W)
      INTEGER, INTENT(IN)  :: N0
      REAL*8,INTENT(IN)    :: ALPHA,BETA,GAMMA,T,H
      REAL*8,INTENT(INOUT) :: W(N0)
      INTEGER              :: I
      REAL*8               :: G(N0),W1(N0)
      REAL*8               :: K1(N0), K2(N0), K3(N0), K4(N0)
      DO I=1,N0
        W1(I)=W(I)
      END DO
      CALL PROBLEM(N0,ALPHA,BETA,GAMMA,T,W,G)
      DO I=1,N0
        K1(I)=H*G(I)
```

```
      END DO
      DO I=1,N0
        W(I)=W1(I)+0.5D+00*K1(I)
      END DO
      CALL PROBLEM(N0,ALPHA,BETA,GAMMA,T,W,G)
      DO I=1,N0
        K2(I)=H*G(I)
      END DO
      DO I=1,N0
        W(I)=W1(I)+0.5D+00*K2(I)
      END DO
      CALL PROBLEM(N0,ALPHA,BETA,GAMMA,T,W,G)
      DO I=1,N0
        K3(I)=H*G(I)
      END DO
      DO I=1,N0
        W(I)=W1(I)+K3(I)
      END DO
      CALL PROBLEM(N0,ALPHA,BETA,GAMMA,T,W,G)
      DO I=1,N0
        K4(I)=H*G(I)
      END DO

      DO I=1,N0
        W(I)=W1(I)+(
K1(I)+2.0D+00*K2(I)+2.0D+00*K3(I)+K4(I) )/6.0D+00
      END DO
    END SUBROUTINE RUNGE
```

Завдання.

1) Знайти розв'язок диференціального рівняния $y' = 2xy^2$ з початковою умовою $y_{|x=0} = 0,25$ на відрізку $[0,1]$ з кроком $h = 0,2$ за допомогою методу Рунге-Кутта 2-го порядку (метод предиктор-коректор).

2) Знайти розв'язок диференціального рівняния $y'' + 4y = \cos(3x)$ з початковою умовою $y(0) = 0,8,\ y'(0) = 2$

на відрізку $[0,1]$ з кроком $h=0,1$ за допомогою метода Рунге-Кутта 4-го порядку.

3) Знайти розв'язок диференціального рівняния $y'(x)=-y(x),\ y(0)=1$ з кроком $h=0,1$ за допомогою метода Рунге-Кутта 4-го порядку.

4) Знайти розв'язок диференціального рівняния $y'(x)=-y(x),\ y(0)=1$ з кроком $h=0,1$ за допомогою метода Рунге-Кутта 2-го порядку (предиктор-коректор).

5.5. Методи оптимізації. Метод градієнтного спуску.

Задача полягає у відшуканні мінімуму функції двох змінних $f(x,y)$ (слід зазначити, що якщо необхідно знайти максимум деякої функції $F(x,y)$, те ця задача зводиться до пошуку мінімуму функції $f(x,y)=-F(x,y)$).

Більшість чисельних методів полягає у відшуканні деякої послідовності (x_0,y_0), $(x_1,y_1),..,(x_k,y_k)$, яка при $k\to\infty$ збігається до точки мінімуму (x^*,y^*). Якщо при цьому виконується $f(x_0,y_0)>f(x_1,y_1)>...>f(x_k,y_k)$, тобто значення функції монотонно спадають при збільшенні k, то такий метод називається методом спуску.

Відомо, що вектор градієнта функції

$$\overline{grad}\, f(x,y)=\left(\frac{\partial f}{\partial x},\frac{\partial f}{\partial y}\right)$$

спрямований убік найбільшого зростання функції $f(x,y)$. Тому як напрямок руху можна прийняти протилежний градієнту напрямок (антиградієнт), тобто координати точок обчислюються за формулами

$$x_{k+1} = x_k - \alpha_k \frac{\partial f(x_k, y_k)}{\partial x},$$
$$y_{k+1} = y_k - \alpha_k \frac{\partial f(x_k, y_k)}{\partial y}. \qquad (1)$$

Вибір величини α_k, з якою пов'язана довжина k-го кроку, у загальному випадку є складною задачею. Якщо α_k – мала величина, то рух буде занадто повільним і буде потребувати значного обсягу обчислень. Якщо α_k – велике, то існує імовірність перескочити точку мінімуму і вийти на протилежний схил функції. При цьому можливо порушення вимоги монотонного спадання послідовності $f(x_k, y_k)$ і з'являється небезпека Вациклення, тобто коливання послідовності (x_k, y_k) у деякому околі точки мінімуму (x^*, y^*) без наближення до неї.

Існує декілька різних способів вибору α_k. Ми розглянемо різновид методу із дробленням кроку. Для цього задається початкове наближення (x_0, y_0) і початкове значення α_0 (наприклад, $x_0 = y_0 = 0$, $\alpha_0 = 1$). Обчислення x_1, y_1 і всіх наступних x_{k+1}, y_{k+1} виконується по формулі (1). При цьому, якщо виявиться, що $f(x_{k+1}, y_{k+1}) > f(x_k, y_k)$, то величина α_k зменшується в два рази й обчислення x_{k+1}, y_{k+1} повторюється від точки (x_k, y_k) з новим значенням α_k. Якщо ж значення функції зменшується, то величина $\alpha_k = \alpha_{k-1}$.

За критерій закінчення обчислень приймається нерівність

$$\left|\overline{\text{grad}}\, f(x,y)\right| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} < \varepsilon$$

або одночасне виконання двох нерівностей

$$\left|\frac{\partial f}{\partial x}\right| < \frac{\varepsilon}{2}, \qquad \left|\frac{\partial f}{\partial y}\right| < \frac{\varepsilon}{2}$$

Порядок виконання роботи на ЕОМ

1. Скласти програму мінімізації $f(x,y)$ методом градієнтного спуску.
2. Задати вхідні дані відповідно до номера варіанта.
3. Провести обчислення на ЕОМ.
4. Написати звіт, який повинен містити результати пунктів 1-3, а також коментар ходу обчислень із поясненнями результатів.

Завдання.

Мінімізувати функцію $f(x,y) = ax + by + e^{cx^2+dy^2}$ методом градіїнтного спуску з точністю до $\varepsilon = 10^{-4}$. Коефіцієнти вибрати з таблиці. Представити два варіанти розрахунку: з коефіцієнтами з верхньої й нижньої частин таблиці. Пояснити різницю в роботі алгоритму.

Варіант	*a*	*b*	*c*	*d*
1	1	-1.4	0.01	0.11
2	2	-1.3	0.04	0.12
3	3	-1.2	0.02	0.13
4	4	-1.1	0.16	0.14

5	5	-1.0	0.25	0.15
6	6	-0.9	0.36	0.16
7	7	-0.8	0.49	0.17
8	8	-0.7	0.64	0.18
9	9	-0.6	0.80	0.19
10	10	-0.5	0.94	0.20
11	11	-0.4	1.00	0.21
12	12	-0.3	1.21	0.22
13	13	-0.2	1.44	0.23
14	14	-0.1	1.69	0.24
15	15	0.0	1.96	0.25
16	16	0.1	1.99	0.26
17	17	0.2	2.56	0.27
18	18	0.3	2.89	0.28
19	19	0.4	3.24	0.29
20	20	0.5	3.81	0.30
21	21	0.6	4.00	0.31
22	22	0.7	5.02	0.32
23	23	0.8	4.84	0.33
24	24	0.9	5.29	0.34
25	25	1.0	5.76	0.35
26	26	1.1	7.25	0.36

27	27	1.2	6.76	0.37
28	28	1.3	5.98	0.38
29	29	1.4	7.29	0.39
30	30	1.5	8.41	0.40

Список рекомендованої літератури.

1.Бартеньев О.В. ФОРТРАН для студентов. — М.: “Диалог МИФИ”, 1999. — 400 с.

2. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. — М.: “Бином”, 2003. — 630 с.

3. Берков Н.А., Беркова Н.Н. Алгоритмический язык Фортран 90. Учебное пособие — М.: МГИУ, 1998. — 96 с.

4. В.Ф. Звягин, Н,А.Яньшина, В.Н.Голыничев. Практикум по современному фортрану в куре информатики. Учебное пособие — С.-П.: ИТМО, 2010. — 134 с.

5. Мак-Кракен Д., Дорн У. Численные методы и программирование на фортране. – 2-е изд.: Пер. с англ. — М.: “Мир”, 1977. — 584 с.

6. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. – Томск: МП “Раско”, 1991. — 272 с.

7. Рыжиков Ю.И. Современный Фортран. — СПб.: “Корона принт”, 2004. — 288 с.