Міністерство освіти і науки України
Донбаська державна машинобудівна академія (ДДМА)

Л. В. Васильєва, С. В. Малигіна, О. В. Бережна

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ, ОБРОБЛЕННЯ МЕДИЧНИХ ДАНИХ

Навчальний посібник

для здобувачів вищої освіти

спеціальності 122 «Комп'ютерні науки»

Затверджено
на засіданні вченої ради
Протокол № 11 від 30.06.2022

Краматорськ
ДДМА
2022

УДК 004.421.2:61(075.8)
В 19

Рецензенти:

Вовна О. В., доктор технічних наук, професор, академік Академії Метрології України, завідувач кафедри електронної техніки;

Разживін О. В., канд. техн. наук, доцент, доцент кафедри автоматизації виробничих процесів Донбаської державної машинобудівної академії.

Васильєва, Л. В.

В 19 Алгоритмізація та програмування, оброблення медичних даних : навчальний посібник для здобувачів вищої освіти спеціальності 122 «Комп'ютерні науки» / Л. В. Васильєва, С. В. Малигіна, О. В. Бережна. – Електрон. дані. – Краматорськ : ДДМА, 2022. – 1 електрон. опт. диск (CD-ROM); 12 см. – Назва з тит. екрана.
ISBN 978-617-7889-26-6.

Розглянуто різновиди алгоритмів і теоретичні відомості мови програмування С, аспекти застосування програмування для оброблення медичних даних. Містить лекційні матеріали, згідно з програмою курсу, приклади, контрольні питання для самоперевірки знань студентів, перелік літератури.

УДК 004.421.2:61(075.8)

ISBN 978-617-7889-26-6

ЗМІСТ

ВСТУП

Найбільш поширеною мовою програмування упродовж кількох останніх десятиріч, поза жодним сумнівом, є мова С, на підставі якої «виросло» багато сучасних мов програмування і програмних середовищ. Цьому сприяли такі її властивості, як лаконічність, потужність, гнучкість, мобільність, можливість доступу до всіх функціональних засобів системи. Програмувати на С можна як для Windows, так і для Unix, причому для кожної з операційних систем існує значна кількість засобів розробляння: від компіляторів до потужних інтерактивних середовищ, як, приміром, Borland C++ Builder, Microsoft Visual C++ чи Visual Studio.NET. Програми на мові С мають низькі вимоги до апаратної частини обчислювальних систем. Проте зараз мова С широко використовується для розробки програмного забезпечення, будучи одним з найпопулярніших мов програмування. Галузь застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також ігор. Більше того, мову С можна використовувати і для створення веб-сайтів через технологію CGI (Common Gateway Interface – загальний шлюзовий інтерфейс).

У посібнику розглянуто основи алгоритмізації медичних задач, формальна логіка у вирішенні задач діагностики, лікування та профілактики захворювань, оброблення масивів даних.

Посібник може використовуватись як джерело прикладів, вправ і програм для оброблення математичних даних, інженерних розрахунків, для багатовимірного аналізу медичних даних та їхнього оперативного аналітичного оброблення з візуалізацією результатів аналізу за допомогою сучасних інформаційних технологій.

1 ОСНОВИ АЛГОРИТМІЗАЦІЇ МЕДИЧНИХ ЗАДАЧ. АЛГОРИТМИ ТА ЇХНІ ВЛАСТИВОСТІ. СПОСОБИ ПОДАННЯ АЛГОРИТМІВ. ТИПИ АЛГОРИТМІВ. ЛІНІЙНИЙ, РОЗГАЛУЖЕНИЙ І ЦИКЛІЧНИЙ ОБЧИСЛЮВАЛЬНІ ПРОЦЕСИ

1.1 Основні принципи розроблення алгоритмів

При постановці задачі першорядна увага має бути приділена з'ясуванню кінцевої мети і виробленню загального підходу до досліджуваної проблеми; вивченню загальних властивостей розглянутого фізичного явища чи об'єкта; аналізу можливостей конкретної ЕОМ і даної системи програмування. Правильно сформулювати задачу іноді не менш складно, чим її вирішити.

Формалізація, як правило, зводиться до побудови математичної моделі розглянутого явища, коли в результаті аналізу суті задачі визначаються обсяг і специфіка вихідних даних, уводиться система умовних позначок, установлюється приналежність задачі, що розв'язується, до одного з відомих класів задач і вибирається відповідний математичний апарат.

Формалізація – процес подання інформації про об'єкт у вигляді алгоритму. У результаті аналізу задачі визначається специфіка даних, вводиться система умовних позначень, встановлюється приналежність її до одного з класів задач (наприклад, математичні, фізичні, медичні тощо).

Формалізована медико-біологічна задача повинна бути ***алгоритмізованою***. Під алгоритмізацією розуміють метод опису систем або процесів шляхом створення алгоритмів їх функціонування.

Під ***алгоритмом*** зазвичай розуміють правило, що вказує дії, у результаті виконання яких приходимо до шуканого результату. Таку послідовність дій називають алгоритмічним процесом, а кожну дію – його кроком. Етап алгоритмізації в загальному випадку настає лише тоді, коли зро-

зуміла постановка задачі, коли існує чітка формальна модель, у рамках якої буде, власне, відбуватися розв'язання задачі. З цієї точки зору процес підготовки задачі передбачає:

- постановку задачі – визначення її змісту та вихідних даних;
- розробку алгоритму розв'язання – вибір методу розв'язування та опис послідовності дій;
- представлення алгоритму розв'язання – побудова структурної схеми алгоритму.

Алгоритми та їхні властивості

У IX ст. узбецький математик Мухаммед, уродженець Хорезма (арабською «альХорезмі»), розробив правила виконання чотирьох арифметичних дій над числами в десятковій системі числення. Множину цих правил назвали алгоритмом (algorithmi – від латинського написання імені альХорезмі), а потім словом «алгоритм» почали позначати сукупність правил певного виду, а не тільки правил виконання арифметичних дій.

Алгоритм – це упорядкований скінчений набір чітко визначених правил для розв'язування задач за скінчену кількість кроків.

Будь-який алгоритм повинен мати такі основні властивості:

Визначеність. Алгоритм не повинен містити вказівок, зміст котрих може бути сприйнятий неоднозначно. Крім того, після виконання чергової вказівки алгоритму не має виникати ніяких суперечностей відносного того, яка вказівка буде виконуватися наступною. Інакше кажучи, при виконанні алгоритму ніколи не повинна з'являтися потреба у прийнятті будь-яких рішень, котрі непередбачені укладачем алгоритму.

Масовість. Алгоритм складається не для розв'язання однієї конкретної задачі, а для цілого класу задач одного типу. В простому випадку ця варіативність алгоритму забезпечує можливість використання різних допустимих вихідних даних.

Дискретність. Процес, який описується алгоритмом, має бути поділений на послідовність окремих дій. Описання, що при цьому виникає, являє собою послідовність чітко відокремлених одна від однієї вказівок, котрі утворюють дискретну структуру алгоритмічного процесу – лише виконавши вимоги однієї вказівки, можна перейти до наступної.

Результативність – обов'язкова властивість алгоритмів. Її суть полягає у тому, що при точному виконанні всіх вказівок алгоритму процес прийняття рішення (отримання результату) повинен закінчитися через скінчену кількість кроків і при цьому має бути отримана відповідь на поставлені в задачі питання.

Способи подання алгоритмів

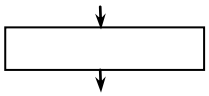
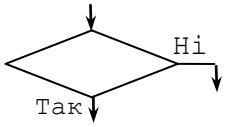
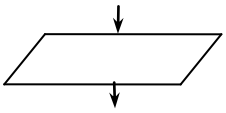
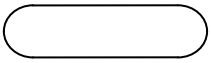
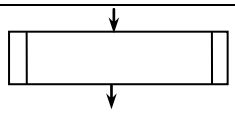
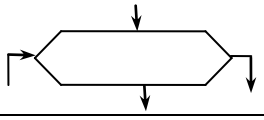
Існує кілька способів подання алгоритмів: словесний, символічний, графічний. Словесний спосіб полягає в описуванні алгоритму в термінах української мови. Даний спосіб застосовується рідко, оскільки запис при цьому досить громіздкий і можуть виникнути суперечливі тлумачення алгоритму.

Символічний спосіб полягає в записі алгоритму за допомогою умовних символів. Даний спосіб подання алгоритму робить запис алгоритму дуже стислим, і не наочним.

Графічний спосіб – зображення алгоритму у вигляді структурної схеми, котра складається з окремих блоків. Цей спосіб подання алгоритму є найбільш зручним і наочним.

Розробка алгоритму полягає в розкладанні обчислювального процесу на можливі складові частини, установленні у порядку їхнього проходження. У практиці програмування широке поширення одержали схеми алгоритмів. Схема – це послідовність типових структурних блоків, що показує порядок виконання визначених функцій, і зв'язки між ними. Усередині блоків дається інформація, що характеризує виконувані ними дії. Деякі, найбільше часто уживані блоки і пояснення до них, наведені в таблиці 1.1.

Таблиця 1.1 – Умовні графічні позначення блок-схем

Назва символу	**Символ**	**Відображувана функція**
1 Блок обчислень		Обчислювальна дія чи послідовність обчислювальних дій
2 Логічний блок		Вибір напрямку виконання алгоритму в залежності від деяких умов (умови)
3 Увід-вивід		Загальне позначення вводу чи виводу даних (поза залежністю від фізичного носія)
4 Початок-кінець		Початок чи кінець програми, останов, вхід чи вихід у підпрограмах
5 Визначений процес (підпрограма)		Обчислення за стандартною підпрограмою чи підпрограмою користувача
6 Блок модифікації (заголовок циклу)		Виконання дій, що змінюють пункти алгоритму
7 Міжсторінковий з’єднувач		Указівка зв’язку між частинами схеми, розташованими на різних сторінках

Алгоритм **лінійної структури** – це алгоритм, у якому блоки виконуються один за одним. Такий порядок виконання блоків називається природним (рис. 1.1).

Приклад 1. Обчислити $y = 2\cos^2 x$, де $x = 2\ln a$; $a = 6{,}7$.
Порядок роботи:

Крок 1. Уводимо a.

Крок 2. Обчислюємо $x = 2\ln a$.

Крок 3. Обчислюємо $y = 2\cos^2 x$.

Крок 4. Друкуємо y.

Крок 5. Кінець алгоритму

Блок-схема

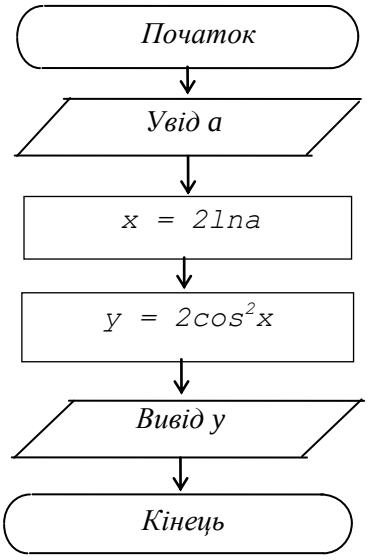


Рисунок 1.1 – Блок-схема алгоритму лінійної структури

Алгоритм **структури, що розгалужується,** – це алгоритм, у якому передбачене розгалуження виконуваної послідовності дій у залежності від результату перевірки якої-небудь умови (рис. 1.2).

Приклад 2. Обчислити корені квадратного рівняння $ax^2 + bx + c = 0$ за умови $d = b^2 - 4ac \geq 0$ за формулою $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

У противному випадку, тобто при $d < 0$, передбачити вивід повідомлення *«Дійсних коренів немає».*

Порядок роботи:

Крок 1. Уводимо a, b, c.

Крок 2. Обчислюємо $d = b^2 - 4ac$.

Крок 3. Якщо $d < 0$, виводимо повідомлення «Дійсних коренів немає», перехід на крок 6.

Крок 4. Обчислюємо $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

Крок 5. Виводимо x_1, x_2.

Крок 6. Кінець алгоритму.

Блок-схема

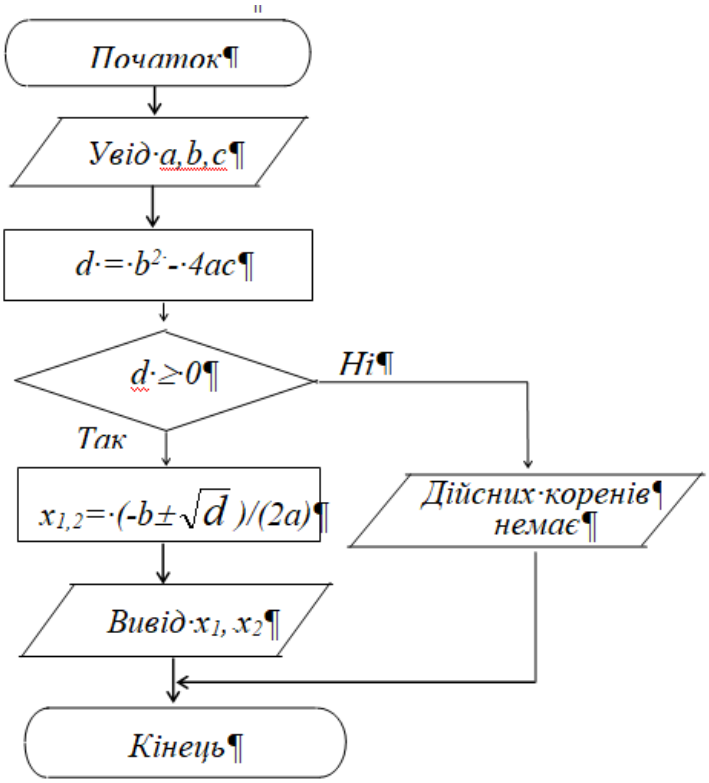


Рисунок 1.2 – Блок-схема алгоритму структури, що розгалужується

Алгоритм **циклічної** структури – це алгоритм, у якому передбачене кількаразове виконання однієї і тієї ж послідовності дій, називаної циклом. Структура циклу: спочатку встановлюємо початкові значення всім змінним циклу, тобто визначаємо їхній стан до першого виконання операцій. Потім описуємо операції, виконувані багаторазово, тобто тіло циклу, далі робимо іншою змінну, визначаючу кількість повторів циклу, тобто параметр циклу. Завершується алгоритм умовою виходу з циклу (рис. 1.3).

Приклад 3. Обчислити $y = sin(3x)/x$ при $1 \le x \le 100$ із кроком 0,5.

Порядок роботи:

Крок 1. Задаємо початкове значення $x = 1$.

Крок 2. Поки $x \le 100$, виконуємо кроки 3–6, інакше – крок 7.

Крок 3. Обчислюємо $y = sin(3x)/x$..

Крок 4. Виводимо x, y.

Крок 5. Збільшуємо значення x на крок: $x = x + 0,5$.

Крок 6. Повертаємося на крок 2.

Крок 7. Кінець алгоритму.

Блок-схема

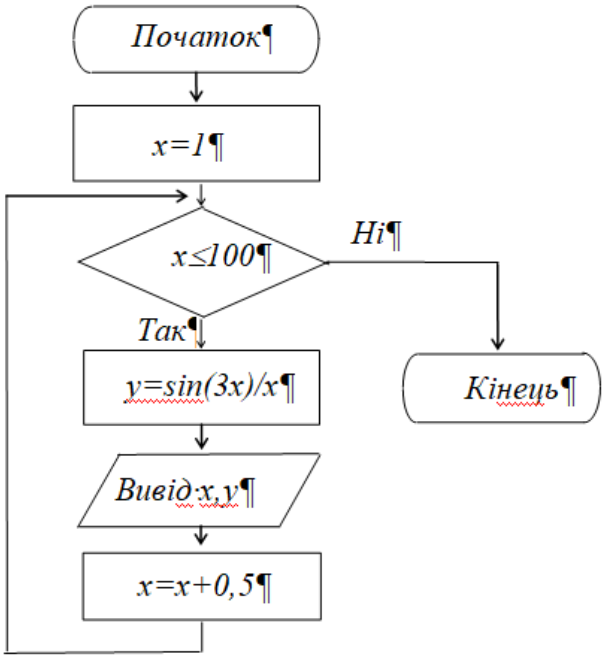


Рисунок 1.3 – Блок-схема алгоритму циклічної структури

1.2 Лабораторна робота № 1. Алгоритмізація лінійного, розгалуженого та циклічного обчислюваних процесів

Мета роботи: виробити практичні навички в побудові блок-схем на лінійний, розгалужений і циклічний обчислювальні процеси.

Звіт повинен містити:

1) мету роботи;

2) умову задачі;

3) блок-схему алгоритму розв’язання задачі;

4) короткі висновки з роботи.

Лабораторна робота містить 4 завдання.

Завдання 1. Накреслити блок-схему для обчислення *h*, узявши значення функцій з таблиці 1.2.

Таблиця 1.2 – Варіанти завдань

Вар.	***H***	***a***	***b***	***c***	***x***
1	a^2+b^2-6c	x^2-e^{-x}	$lnx+\sqrt{x}$	cos^2x+x^5	5,4
2	$c^2+8b+10a$	$sin^2x+x^{1/4}$	$tgx-8x^3$	$x^4+2sinx^2$	1,2
3	$3a^2+4b-8$	$3x-2cos^3x$	$lnx+2e^x$	$x^{1/3}+4x-1$	0,3
4	a^3+b^2-8c	sin^3x+x^4	$\sqrt{x}-lnx$	$4x-5x^3$	1,7
5	$6b^3+4c-2$	$tgx+e^{2x}$	x^2-6x^3	$1/x-2lnx$	4,1
6	$a^2+b^2+c^2$	$e^x+e^{2x}+4$	$x-sin^3x$	x^2/cos^3x	2,4
7	$5b^3-2a+c$	$tgx-2x$	$\sqrt{x}-sinx$	$x^3/7$	5,5
8	$4a^2+5b^2$	$cosx+2x$	$x^4-2x/5$	$2x-5$	4,6
9	$3ab-4c$	sin^2x+5	$cosx^5$	$x^{1/3}+tgx$	1,6
10	c^2+5a^3-b	cos^3x-6x	$-4x^3+lnx$	$e^{2x}+4cosx$	4,6
11	$2a+4c-b^4$	e^x-2lnx	$2x-5/x$	x^5-2lnx	3,9
12	$a^2+b^2+c^2$	$2/x+x^3$	lnx^2-4x	$tgx-sin2x$	4,1
13	$(a+b)^2$	$lnx+2e^x$	$tgx+e^{2x}$	x^2-e^{-x}	3,4
14	$2ac-3cb$	$1/x-2lnx$	$cosx+2x$	$sin^2x+x^{1/4}$	1,9
15	$5c+2a^4$	x^2-2/x	$(2-x)/6$	cos^3x-2x	2,3
16	$a+b+c$	$lnx/2x$	x^3-4x	$tgx-2x$	4,2
17	$2a+3b+4c$	x^2+x^3	$lnx-x^4$	$cos^2(x-4)$	2,8
18	$a^2+b^3+c^4$	$sin^2x+x^{1/4}$	x^3+4x	e^x+2lnx	1,3
19	$a+2b+3c$	$2x-x^{1/4}$	$\sqrt{x}-2cosx$	$tgx-4x$	3,1
20	$2(a+b)-c^4$	$(x^3-x/2)^3$	$lnx-e^{2x}$	$\sqrt{x+x^3}$	2,4
21	c^2-b^3	$2x+sinx^4$	$sin(x-lnx)$	lnx^2+2x	1,1
22	$3a-4cb$	$2cosx^3$	$tgx/4$	$x/5$	3,1
23	c^5-2ab	$½sin^3x$	sin^6x/x^3	$x-4sin2x$	1,8
24	$6a+3b^3+c$	$cosx^x+2x$	$sin2x+tgx$	$lnx-e^{-x}$	2,1
25	$4abc$	x^x-sinx^3	$x/2-x^5$	$2x-sin^3x$	4,1

Завдання 2. Накреслити блок-схему для обчислення *y = f(x)*, де

$$x=\begin{cases} f_1(z), & \text{якщо } z<0; \\ f_2(z), & \text{якщо } 0\le z\le 8; \\ f_3(z), & \text{якщо } z>8; \end{cases} \qquad z = cos(c).$$

Значення функцій наведені в таблиці 1.3.

Таблиця 1.3 – Варіанти завдань

Вар.	$F(x)$	$f_1(z)$	$f_2(z)$	$f_3(z)$	c
1	x^2+8x-6	z^3-3z^2	$zln(z)$	e^z-e^{-z}	5,1
2	x^3lnx^2	$e^{-z}+3z$	$ln\|z\|$	$cosz+z^2$	5,4
3	$x^{1/4}+sinx$	$2z-ln\|z\|$	$tgz-2z$	sin^3z	4,1
4	$x^4+2sinx^2$	$sinz+tgz$	cos^3z+3/z	z^2+lnz^2	3,2
5	$cosx^3$	$z^2+2sinz$	$lnz+2z$	e^z+1/z	4,7
6	$sinx+2lnx$	$2z+tgz$	lnz^4+2z	$cosz+2z$	1,3
7	sin^4x^2	$sinz^2-z^3$	$\sqrt{\|cosz\|}$	$2sinz^2$	1,6
8	$tgx-4x^3$	$1/cos^2z$	$z-ln\|z\|$	z^3+sinz	1,5
9	$ln(x)-e^{2x}$	z^2+e^z	cos^4z/z^3	$tg(z+1/z)$	2,7
10	$2x-ln(x)$	$2cosz+1/z$	$z^3-2ln\|z\|$	$tg2z+z^3$	3,8
11	$3x-sin(x)$	$3tg^3z$	$1/cos^4z$	$e^{2z}+sinz$	1,6
12	$4x^2+cosx$	$3z/sinz$	$z^2+2sinz$	$2z-ln\|z\|$	2,4
13	$\sqrt{x}+cosx$	z^2+lnz^2	e^z+1/z	z^4-sinz	4,1
14	$cosx^4+2x$	$tg(z+1/z)$	$e^{2z}+sinz$	$cosz^{1/5}$	3,8
15	sin^4x+2x	z^2+lnz^2	cos^3z+3/z	$cos(\pi/4)-z$	5,8
16	$3ln(x^2+5)$	z^4-lnz	$sinz+tgz$	$sinz+cosz$	3,5
17	$ln(1/x)$	$zsin^2z-8$	$sinz^{0,8}$	$\sqrt{ze^z}-2,5$	2,3
18	$e^{2x}+4x$	$cos(\pi/4)-z$	$1/(e^z+1)$	$tg(z+3)$	4,1
19	$cosx^4+x/2$	$sin(z+30°)$	$cos(\pi z/6)$	$e^{(z-2)}$	3,2
20	$2tgx+e^x$	$z+cos(\pi+z)$	$z^3+z^{1/3}$	z^4-lnz	2,8
21	$cos(x)^4+x/2$	$sinz^{0,8}$	$cos(\pi/4)-z$	$z/sinz^{1/5}$	3,7
22	$cos^2x/3$	$z^2+ln(z+4)$	$e^{(z-5)}+sinz$	$\sqrt{z^4+tgz}$	2,2
23	$1/tgx^4$	$e^{-4z+2}+z^2$	$cos(z^{1/3}+2)$	$sin(\pi+4z^2)$	5,6
24	$e^{2x}-x^3$	$tg(z^2+\sqrt{z})$	$ln(sinz+5)$	z^4+z^2-cosz	3,4
25	$tgx-2lnx$	$arcsin(z+3)$	z^3-z^2+cosz	$ln(z^3+4z)$	2,5

Завдання 3. Накреслити блок-схему для обчислення таблиці значень функції для значень аргументу x в інтервалі від *Xп* до *Xк* з кроком *ΔX*. Вихідні дані наведені в таблиці 1.4.

Таблиця 1.4 – Варіанти завдань

Вар.	***F(x)***	***a***	***Xн***	***Xк***	***ΔX***
1	$ax+\sqrt{\lvert x\rvert}$	π/4	-π	π	π/3
2	$\frac{\sin^2 x}{\sqrt{x}+ax}$	4,1	1,2	3,6	0,2
3	$\frac{e^{ax}+5}{1+\cos^2 x}$	2,8	1,4	4,2	0,3
4	$e^{ax}+\cos^2 x$	3,9	2,8	4,6	0,2
5	$\frac{e^{-x}(1+ax)}{\ln^2 x}$	2,4	0,7	3,8	0,2
6	$\frac{e^{-ax}}{\ln(x+a)}$	π	0,1	2π	π/6
7	ax^2+e^x	2,1	0,8	3,6	0,2
8	$\frac{e^{-(a+x)}}{a+\cos^3(ax)}$	5,4	2,3	8,9	0,4
9	$\sqrt{1+e^{ax}}$	-1	-4,2	3,8	0,5
10	$\frac{a-e^{a-x}}{\ln^2 x}$	14,2	11,6	1,6	0,2
11	$\frac{a+\sin^2(ax)}{e^{-x/2}}$	1,1	0,2	1,8	0,2
12	$\frac{atg^2 x}{a+0,7x}$	5,4	2,2	7,3	0,3
13	$\frac{a-\sqrt{ax}}{1+\cos^2 x}$	2,5	1,9	3,8	0,2
14	$\frac{\sqrt{a\ln x}}{1+tg^2(ax)}$	5,1	3,3	6,9	0,3
15	$\frac{e^{ax}+a^x}{\sqrt{1+e^{ax}}}$	0,7	0,6	0,9	0,05
16	$\frac{1+\sin^2(a+x^2)}{\sqrt[3]{a+x^2}}$	3,8	1,2	5,3	0,4
17	$\frac{10x-e^{ax}}{ax+2}$	2,7	1,8	4,2	0,3
18	$ax+\sqrt{\frac{x}{a}}$	2,1	0,4	3,9	0,2
19	$\frac{\sqrt{\lvert\sin(ax)\rvert}}{1+x^3}$	3,4	0,5	4,6	0,5
20	$a\sin(x^3)+x$	2,8	1,6	4,8	0,3
21	$x^4\sin(ax)$	4,2	2,8	5,6	0,2
22	$ax+tg(ax)$	2,6	0,5	3,5	0,5
23	$\sqrt{ax^2+\sin x}$	6,3	2,5	7,5	0,5
24	$e^{\sqrt{x}}(1+ae^x)$	5	1	10	0,5
25	$a\sqrt{e^x+\pi}$	π	0	2π	π/4

Завдання 4. Знайти суму ряду $y = \sum \frac{f_1}{f_2}$, де $a \le x \le b$, $\Delta x = c$.

Варіанти завдань наведені в таблиці 1.5.

Таблиця 1.5 – Варіанти завдань

Вар.	F_1	f_2	**a**	**b**	**c**
1	$3x-1$	$e^{-1/x}+x/(x+1)$	3	5	0,5
2	x^3-3x^2	x^4+2x^2+3	1	3	0,2
3	$e^{-x}+4x$	$\sqrt{(1+x^3+4x^2)}$	0,6	4,2	0,3
4	$\sin^2(x+4x^3)$	$(x+2x^3)\sqrt{x}$	0,5	4,8	0,2
5	$x\sin x^3-\ln 2x$	$arctgx/4+e^{-x+2}$	2	6,3	0,4
6	$x^4-\cos x$	$tgx+2x$	1	5	0,5
7	$2x+\sin^2 x$	$\sqrt{(2x+\ln x)}$	5	8	0,3
8	$\ln(4x+8)$	$e^{-x}+\sin 2x$	1	4	0,2
9	$x^3\ln(2x)$	$4x^2+6x^3-2$	0,5	6	0,3
10	$x^2+\sin^3 x$	$\cos 3x+e^{-2x}$	-2	3	0,4
11	xe^{-x}	$\sin 4x+x^3$	1,5	5	0,3
12	$\sqrt{x^2+1\|}$	$arctgx/5+2x$	0,6	4	0,2
13	$x^2/(3x+2)$	$\sin^2(\pi x+1)$	0,5	5,2	0,3
14	$\sqrt{2+x^3}$	$3^x/(x-2)$	1,2	6,3	0,4
15	$x^{3x+1}+8x$	$\|x-8\|+\sin x$	4	7,5	0,3
16	x^4+e^{x+3}	$x\,arctg(x/3)$	2	6,4	0,2
17	$\ln^2(x+4)$	$\sin^3(x/5)$	1	6,8	0,3
18	$e^{x-2}+x^3$	$x-\ln\|x-1\|$	0	4	0,4
19	$2\cos(x+3)$	$4x^2/(3+x^3)$	2	5	0,3
20	$\sqrt{1+x^4})$	$tg^2(x+4)-e^{-x}$	1	6	0,4
21	$3+2\sin^2(x-3)$	$4+x/10$	2	7	0,5
22	$\ln(1(1+2^x)$	$\sin^2(4x+1)$	1,5	6,8	0,4
23	$\sqrt{\|x\|}+e^{-x}$	$5arctg(4x)$	2	7	0,5
24	$\arcsin(x+2)$	$3(x-4)/(x^2+1)$	3	8	0,2
25	$e^{\|x+2\|}$	$\ln^2(x+4)$	-2	6	0,3

2 СКЛАДАННЯ Й НАЛАГОДЖЕННЯ ПРОСТИХ ПРОГРАМ

2.1 Програмування лінійного обчислювального процесу

2.1.1 Функція форматованого виведення

Для виведення інформації в програмах використовується функція **printf ().** Вона переводить дані з внутрішнього коду в символьне уявлення і виводить отримані зображення символів результатів на екран. При цьому у програміста є можливість форматувати дані, тобто впливати на їх подання на екрані.

Можливість форматування умовно відзначена в самому імені функції за допомогою літери ***f*** в кінці її назви (*print formatted*).

Оператор виклику функції **printf ()** записується так:

printf *(«форматний рядок «, список _аргументов);*

Форматний рядок обмежений подвійними лапками і може включати довільний текст, керуючі символи і специфікації перетворення даних. Список аргументів (з попередньої коми) може бути відсутнім. Наприклад:

```
#include <stdio.h>
void main( )
{ printf («\n Hello world!\n»);
}
```

Директива *#include <stdio.h>* включає в текст програми опис (прототип) бібліотечної функції ***printf ()***. Якщо видалити з тексту програми цю препроцесорну директиву, то з’являться повідомлення про помилки і виконаний код програми не буде створено. Серед параметрів функції ***printf ()*** у цьому прикладі є тільки форматний рядок, а список аргументів відсутній.

У форматному рядку два керівних символи ***‘\ n’*** – «новий рядок». Між ними знаходиться текст, який виводиться на екран: *Hello world!*

Перший символ ***‘\ n’*** забезпечує висновок цієї фрази з початку нового рядка. Другий керуючий символ ***‘\ n’*** переведе курсор на початок наступного рядка, де і розпочнеться виведення інших повідомлень (не пов’язаних з програмою) на екран.

Для зображення в програмі відповідних символьних констант використовуються комбінації з декількох символів, що мають графічне представлення. Кожна така комбінація починається з символу **‘\’** (зворотна коса риса – backslash). Такі набори літер, що починаються з символу **‘\’**, у літературі з мови Сі називають керівними послідовностями (таблиця 2.1).

Таблиця 2.1 – Керівні послідовності

Керівні послідовності	***Дія***
‘\n’	переведення рядку
‘\t’	горизонтальна табуляція
‘\r’	повернення каретки (курсору) до початку рядка
‘\\’	зворотна коса риска \
‘\n’’	апостроф (одинарні лапки)
‘\’»	лапки (символ подвійної лапки);
‘\0’	нульовий символ
‘\a’	сигнал-дзвоник
‘\b’	повернення на одну позицію (на один символ)
‘\f’	переведення (прогін) сторінки
‘\v’	вертикальна табуляція
‘\?’	знак питання

Довільний текст (не специфікації перетворення і не керуючі символи) безпосередньо без змін виводиться на екран. Керуючі символи (переклад рядка, табуляція тощо) дозволяють впливати на розміщення інформації, що виводиться на екрані.

Специфікації перетворення даних призначені для управління формою зовнішнього представлення значень аргументів функції ***printf().*** Узагальнений формат специфікації перетворення має вигляд:

% прапорці ширина _поля. Точність модифікатор специфікатор

Серед елементів специфікації перетворення обов'язковими є тільки два – символ ***%*** і *специфікатор*.

У завданнях обчислювального характеру використовують специфікатор:

d – для цілих десяткових чисел (тип **int**);

u – для цілих десяткових чисел без знака (тип **unsigned)**;

f – для дійсних чисел у формі з фіксованою точкою (типи **float** і **double**);

e – для дійсних чисел у формі з плаваючою крапкою (з мантиси і порядком) – для типів **double** і **float**.

До списку аргументів функції ***printf ()*** включають об'єкти, значення яких повинні бути виведені програмою. Ці вирази й їхні окремі випадки – змінні й константи. Кількість аргументів і їхні типи повинні відповідати послідовності специфікацій перетворення в форматної рядку. Наприклад, якщо речова змінна ***summa*** має значення 2102.3, то при такому виконанні функції

printf(«\n summa=%f», summa);

на екран з нового рядка буде виведено:

summa=2102.3

Після виконання операторів

float c, e;

int к;

c=48.3; к=-83; e=16.33;

printf («\nc=%f\tk=%d\te=%e», c, к, e);

на екрані вийде такий рядок:

c=48.299999 к=-83 e=1.63300e+01

Тут використовується керівний символ ***\t*** (табуляція). З його допомогою виводяться значення в рядку результату, відокремлені одне від одного.

Для виведення числових значень у специфікації перетворення вельми корисні «*ширина_поля*» і «*точність*».

Ширина поля – ціле позитивне число, що визначає довжину (в позиціях на екрані) уявлення виведеного значення.

Точність – ціле позитивне число, що визначає кількість цифр у дробовій частині зовнішнього подання дійсного числа (з фіксованою крапкою) або його мантиси (при використанні форми з плаваючою точкою).

Наприклад, після виконання операторів

float c, e;

int к;

c=48.3; к=-83; e=16.33;

printf («\nc=%5.2\tk=%5d\te=%8.2f\te=%11.4e», c, k , e, e);

на екрані вийде такий рядок:

c=48.30 к= -83 e= 16.33 e= 1.6330e+01

В якості модифікаторів в специфікації перетворення використовуються символи:

h – для виведення значень типу **short int***;*

I – для виведення значень типу **long;**

L – для виведення значень типу **long double**.

2.1.2 Введення даних

Для введення даних із клавіатури в програмі будемо використовувати функцію (описана в заголовки *stdio.h*):

scanf (« форматний рядок «, список_аргументів);

Функція ***scanf ()*** виконує «читання» кодів, що вводяться з клавіатури. Це можуть бути як коди видимих символів, так і керуючі коди, що надхо-

дять від допоміжних клавіш і від їх поєднань. Функція ***scanf ()*** сприймає коди, перетворює їх у внутрішній формат і передає програмі. При цьому програміст може впливати на правила інтерпретації вхідних кодів за допомогою специфікацій рядка формату. Можливість форматування умовно відзначена в назві функції за допомогою літери ***f*** в кінці імені.

І форматний рядок, і список аргументів для функції ***scanf ()*** є обов'язковими. Форматний рядок для функції ***scanf ()*** будемо формувати з специфікацій перетворення виду:

*% * ширіна_поля модифікатор специфікатор*

Серед елементів специфікації перетворення обов'язкові тільки **%** і *специфікатор*. Для введення числових даних використовуються ті ж специфікатори, що і для оператора *printf () – d, u, f, e.*

Ширина_поля – ціле позитивне число, що дозволяє визначити, яка кількість байтів (символів) з вхідного потоку відповідає значенням, що вводяться.

В якості модифікаторів використовуються символи:

h – для введення значень типу **short int (hd);**

l – для введення значень типу **long int (ld)** або **double (lf, le**);

L – для введення значень типу **long double (Lf, Le).**

На відміну від функції **printf ()** аргументами для функції **scanf ()** можуть бути тільки адреси об'єктів програми, в окремому випадку – адреси її змінних. Чи не розшифровуючи поняття адреси (адреси і покажчики будуть розглянуті пізніше), нагадаємо, що в мові Сі є спеціальна унарна операція **&** отримання адреси об'єкта:

& Ім'я_ об'єкта

Вираз для отримання адреси змінної буде таким:

& ім'я_змінної

Отже, для позначення адреси перед ім'ям змінної записують символ *&*. Якщо *name* – ім'я змінної, то *& name* – її адреса.

Наприклад, для введення з клавіатури значень змінних *n, z, x* можна записати оператор:

scanf («% d% f% f «, & n, & z, & x);

В даному випадку специфікації перетворення в форматному рядку не містять відомостей про розміри полів і точність значень, що вводяться. Це дозволено і дуже зручно при введенні даних, діапазон значень яких визначений не строго. Якщо змінна *n* описана як ціла, *z* і *x* – як речові типу *float*, то після читання з клавіатури послідовності символів 18 18 -0.431 змінна *n* отримає значення 18, *z* – значення 18.0, *x* – значення -0.431.

При читанні вхідних даних функція **scanf ()** сприймає як роздільники полів даних «узагальнені пробільні символи» – власне прогалини, символи табуляції, символи нових рядків.

2.1.3 Присвоєння (вираз і оператор)

Символ «=» в мові Сі позначає бінарну операцію, у якій в вираженні повинно бути два операнда: лівий (зазвичай змінна) і правий (зазвичай вираз). Якщо *z* – ім'я змінної, то *z = 2.3 + 5.1* є вираз із значенням 7.4. Одночасно це значення присвоюється і змінної **z**. Тільки в тому випадку, коли в кінці виразу з операцією присвоювання поміщений символ «**;**», цей вислів стає оператором присвоювання. Таким чином, *z = 2.3 + 5.1;* є оператор простого присвоювання змінній *z* значення, рівного 7.4.

Тип і значення виразу з операцією присвоювання визначаються значенням виразу, поміщеного праворуч від знака «=». Однак цей тип може не збігатися з типом змінної з лівої частини виразу. В цьому випадку при визначенні значення змінної виконується перетворення (приведення типів).

Оскільки вираз праворуч від знака «=» може містити, у свою чергу, операцію присвоювання, то в одному операторі присвоювання можна привласнити значення декільком змінним, тобто, організувати «множинне» присвоювання, наприклад:

$$c = x = d = 4.0 + 2.4;$$

Тут значення 6.4 присвоюється змінної *d*, потім 6.4 як значення виразу з операцією присвоювання «*d = 4.0 + 2.4*» присвоюється *x* і, нарешті, 6.4 як значення виразу «**x = d**» присвоюється **c**. Природне обмеження – зліва від знака «=» у кожній з операцій присвоювання може бути тільки зліва допустімий вираз (ім'я змінної).

У мові Сі існує цілий набір «складових операцій присвоювання» (таблиця 2.2).

Таблиця 2.2 – Складові операції присвоювання

Складова операція присвоювання	Пояснення	Приклад операції складеного присвоювання	Еквівалентна операція простого присвоювання
1	*2*	*3*	*4*
*=	*присвоювання після множення:* присвоїти операнду лівої частини множення значень обох операндів	P *= 2;	P = P * 2;
/=	*присвоювання після поділу:* присвоїти операнду лівої частини частка від ділення значення лівого операнда на значення правого	P /= 2.2 – d;	P = P/(2.2 – d);
%=	*присвоювання після поділу по модулю:* присвоїти операнду лівої частини залишок від цілочислового ділення значення лівого операнда на значення правого операнда	N %= 3;	N = N % 3;

Продовження таблиці 2.2

1	2	*3*	*4*
+=	*присвоювання після підсумовування:* присвоїти операнду лівої частини суму значень обох операндів	A += B;	A = A + B;
-=	*присвоювання після вирахування:* присвоїти операнду лівої частини різницю значень лівого і правого операндів.	X -= 4.3 – Z;	X = X – (4.3 – Z);

Кожна із складових операцій присвоювання об'єднує деяку бінарну логічну або арифметичну операцію і власне присвоювання. Операція складеного присвоювання є основою оператора складеного присвоювання:

ім'я змінної op = вираз;

де ***op*** – одна з операцій ***, /,%, +, - , &,** □**, |, «,».** Якщо розглядати конструкцію «***op* =**» як дві операції, то спочатку виконується ***op***, а потім '='.

Перейти від простого оператора присвоювання до складеного можна тільки в тих випадках, коли одна змінна використовується в обох частинах. Більш того, для деяких операцій ця змінна повинна бути обов'язково першим (лівим) операндом. Наприклад, не вдасться замінити складовими наступні прості оператори присвоювання:

a = b / a;

x = z% x.

Як лівого операнда в операціях присвоювання може використовуватися тільки модифікуються іменує вираз, тобто посилання на деяку іменовану область пам'яті, значення якої є змінам.

Ще раз відзначимо, що існують одна проста операція присвоювання і ряд складових операцій (див. таблицю 2.2).

В операціях присвоювання можуть використовуватися стандартні математичні функції (таблиця 2.3). Для цього в програмі необхідно підключити директиву *#include <math.h>*.

Таблиця 2.3 – Запис математичних функцій у мові С++

Математична функція	Назва функції	Функція на мові програмування Сі для аргументу ***x*** типу double
$\arccos x$	арккосинус	*acos(x);*
$\arcsin x$	арксинус	*asin(x);*
arctgx	арктангенс	*atan(x);*
$\sqrt[3]{x}$	кубічний корінь	*pow(x,1.0/3.0)*
e^x	експонента числа	*exp(x);*
$\lvert x \rvert$	модуль числа (абсолютна величина)	*fabs(x);*
$\ln x$	натуральний логарифм	*log(x);*
$\lg x$	логарифм з основою 10	*log10(x);*
x^y	піднесення числа до степеня ***y*** за основою ***x***	*pow(x, y);*
$\sqrt{x}$	квадратний корінь з числа	*sqrt(x);*
$\sin x$	синус	*sin(x);*
$\cos x$	косинус	*cos(x);*
$tg(x)$	тангенс	*tan(x);*
$ctg(x)$	котангенс	*1/tan(x);*

У бібліотеці <*math.h*> визначено також ряд констант, таких як *M_PI* (число π), *M_E* (основа натурального логарифма) і ін.

2.1.4 Приклад програми лінійного обчислювального процесу

Використовуючи розглянуті в цьому розділі оператори, запишемо програму лінійного обчислювального процесу (рис. 2.1)

Приклад 5. Обчислити $y = 2\cos^2(x)$, де $x = 2\ln(a)$; *a=6,7* (див. блок-схему на рис. 1.1).

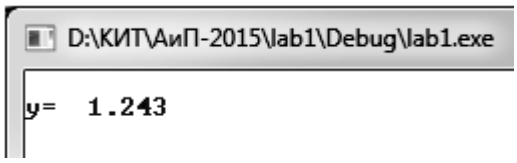
Директива підключення заголовки «*stdio.h*», якщо в програмі треба задіяти консольне введення-виведення за допомогою функцій *printf ()* або *scanf ()*	→	*#include<stdio.h>*
Директива підключення заголовки «*math.h*», якщо в програмі використовуються стандартні математичні функції	→	*#include<math.h>*
Визначення функції *main*, яка не повертає ніякого результату (немає вказівки типу перед *main*) і яка не має аргументів (порожні дужки)	→	*main()*
Початок програми (функції)	→	*{*
	→	*double a=6.7,x,y;*
	→	*x=2*log(a);*
	→	*y=2*cos(x)*cos(x);*
	→	*printf(«\ny=%7.3f»,y);*
Функція *getchar* очікує введення користувача, тим самим «відкладаючи» завершення програми так, що висновок програми буде залишатися на екрані після її виконання	→	*getchar();*
Кінець функції *main*	→	*}*
Результат роботи програми	→	

Рисунок 2.1 – Результат роботи програми

2.2 Лабораторна робота № 2. Розроблення програми лінійного алгоритму

Мета роботи: виробити практичні навички в написанні і налагодженні простих програм.

Завдання. Складіть програму для обчислення функції *b = b (x, y, z), де z = z (x, y).* Вид функції і вихідні дані наведені в таблиці 2.4.

Таблиця 2.4 – Вихідні дані

Вар.	***B(x, y, z)***	***z(x, y)***	***x***	***y***
1	*2*	*3*	*4*	*5*
1	$e^{\lvert x-y\rvert}(tg^2 z)^x$	$\sqrt{sin^2\lvert x\rvert+y}$	-4,52	0,75
2	$\frac{\sqrt{x}sin(\pi x)}{x+e^x y}z$	$\frac{2xy}{x+cosy}$	2,87	0,84
3	$\frac{y-z/(y-x)}{cosx+(y-x)^2}$	$\frac{\sqrt{15y}}{y+ctgx}$	1,82	18,25
4	$ln(\sqrt{x}+\sqrt{y}+2)\cdot z^3$	$\sqrt{x+2y}\cdot sin(x^2)$	5,34	3,85
5	$(arccos x)^2+\lvert x+y\rvert^3$	$\sqrt{x^2+sin y}$	-2,75	-1,42
6	$ln(\sqrt{e^{x-y}})+z^2$	$15/(x+e^y)$	1,54	-3,26
7	$x^{y/x}-\sqrt[3]{\lvert yz\rvert}$	$ln(\sqrt[4]{x^3+y})$	1,82	18,23
8	$y^x+\sqrt[3]{\lvert x\rvert+\lvert y\rvert}\cdot e^z$	$\frac{\sqrt{\lvert 20x\rvert}}{x^2+y^3}$	-0,85	1,25
9	$\frac{ln(x^2+\lvert z\rvert^3)}{sinz+e^{y-x}}$	$\frac{5+\sqrt{\lvert x\rvert}}{cos(y^2)}$	-3,44	5,28
10	$\frac{z^2}{y+x^3}+arcsin(y/5)$	$\frac{\pi x}{cos^2 y+\pi}$	1,58	3,42
11	$\lvert sinx\rvert-2tg^2(\frac{z}{xy})$	$\sqrt{x}\cdot sin y$	0,42	-0,87
12	$ln(y^{\sqrt{\lvert x\rvert}})(z^2-\frac{y}{sinx})$	$\frac{sin(x/y)}{2x^2}$	-15,24	4,67
13	$\sqrt[3]{\lvert\frac{y}{x^2+1}\rvert}+e^{-z}$	$\frac{(y-x)cosx}{x+e^y}$	-4,58	2,32
14	$\frac{\lvert x\rvert+zy-z^3}{\sqrt{6+sin^2 y}}$	$\frac{\pi y}{\pi+e^{x-y}}$	5,48	2,25

Продовження таблиці 2.4

1	2	3	4	5
15	$\sqrt{\lvert y\rvert e^{-(x+y)}}-\cos(z^3)$	$\frac{x+6y}{\sin x+\ln y}$	1,12	0,87
16	$\frac{4y^2e^{2\sin x}}{8z^3+\ln\lvert x\rvert}$	$\frac{x+y\sqrt{x}}{x+10}$	0,27	4,38
17	$\frac{\sqrt{y\ln x}-zx^2}{1+tg^2(x^2)}$	$\frac{e^x\sqrt{x^3+y}}{x-1}$	6,35	7,32
18	$\frac{\ln(y+\sqrt{y+x^2})}{(z+x^2)e^{x/2}}$	$\frac{2x\sqrt{y}}{\sin(x^2)}$	0,42	1,23
19	$\frac{x^3+y}{\sin^2 z+x/5}$	$\frac{\cos^2\pi(2+x)}{4-y^2\sqrt{x}}$	4,32	-0,54
20	$\frac{1+\cos^2(x+z)}{\lvert x^3-2\ln\sqrt{y}\rvert}$	$\frac{x^2+y^2}{e^{x+y}}$	0,83	2,38
21	$\frac{\ln\lvert x\rvert}{\sqrt[3]{\lvert x\rvert+\lvert y\rvert}+tg(z/x)}$	$\frac{1}{x^2+y}$	-0,93	-0,25
22	$2^{-x}\sqrt{y+\sqrt[4]{\lvert z\rvert}}$	$\frac{x+5y}{\sqrt{x}+\ln\lvert y\rvert}$	3,25	4,12
23	$\frac{z^3}{x+y^3/(x+z^2)}$	$\frac{\lvert y+8x\rvert}{\sin x+tg\,y}$	-0,72	-1,42
24	$\frac{x+y(x^2+\cos x)}{y(x-z)+\ln\lvert xz\rvert}$	$\frac{xy}{x^2+5}+\cos^2 y$	3,98	-1,63
25	$\sqrt{e^{(x-1/\sin x)}\sqrt{\lvert y\rvert}}$	$2\sin(\pi x+y)$	3,91	-0,51

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв’язання задачі; програму;

3) рішення задачі (лістинг) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. Чим відрізняються функції printf() або puts() при консольному виведенні інформації?

2. Як здійснюється зчитування з консолі інформації за допомогою функції scanf_s()?

3. Як виводиться на консоль послідовність різних типів даних за допомогою однієї функції printf()?

4. Для яких типів даних використовуються суфікси при ініціалізації змінних?

5. Для яких типів даних використовуються суфікси при ініціалізації змінних?

2.3 Формальна логіка у розв'язанні задач діагностики, лікування та профілактики захворювань. Логічні операції та таблиці істинності. Логічний підхід до діагностики захворювань. Умовні оператори

2.3.1 Логічний підхід до діагностики захворювань

Логічний аналіз застосовується в медицині для діагностики захворювань. Будь-яке захворювання описується комплексом симптомів, характерних для нього, які дають змогу відкинути схожі захворювання. Наявність симптому в хворого позначається символом І, відсутність симптому – Х.

Таким чином, симптоми відіграють роль аргументів, а діагноз захворювань, який може набувати тільки двох значень (або бути істинним для комплексу симптомів, або бути хибним), є логічною функцією цих аргументів. Найбільш простим діагностичним прийомом є пряме зіставлення значень симптомів у хворого і в еталоні захворювання. При повному збігові значень і здійснюється діагностика захворювання. Такий метод застосовується для захворювань, які розвиваються за класичною схемою. Проте досвідчений лікар знає, що дуже рідко патологічні процеси в організмі протікають у відповідності з описами, поданими в підручнику. Більш складним логічним методом є порівняння всіх можливих комбінацій значень симптомів (наприклад, беруть усі комбінації значень у різних сполученнях з трьох симптомів) з даними, які містяться в перевірених історіях хвороби. При порівнянні кожна така комбінація характерна для певної кількості ви-

падків N1 якого-небудь захворювання А і певної кількості випадків N2 інших захворювань. Якщо N1 > N2, то комбінація вважається інформаційною для діагностики захворювання А. Сутність цього логічного методу полягає у визначенні всіх інформаційних, взаємодоповнюючих комбінацій, за якими ставиться діагноз.

2.3.2 Умовний оператор if … else

Умовний оператор може мати скорочену форму:

if (вираз_умова) оператор;

де в якості *вираз_умова* можуть використовуватися: арифметичний вираз, відношення і логічний вираз. Оператор, включений в умовний, виконується тільки в разі істинності (тобто при ненульове значення) *вираз_умова*. Наприклад:

```
if  (x < 0 && x > -10)   x=-x:
```

Крім скороченої форми, є ще і повна форма умовного оператора:

```
if (вираз_умова)
оператор _1;
else
оператор _2;
```

Тут у разі істинності *вираз_умова* виконується тільки оператор_1, при нульовому значенні *вираз_умова* виконується тільки оператор_2. Наприклад:

```
if (x > 0)
b=x;
else
b=-x;
```

Оператор в скороченому варіанті оператора *if*, і *оператор_1* і *оператор_2* в повному операторі *if* можуть бути як окремими, так і складовими операторами.

Наприклад, при вирішенні алгебраїчного рівняння 2-го ступеня $ax^2 + bx + c = 0$ дійсні корені є тільки в разі, якщо дискримінант $(b^2 – 4\ ac)$ не може бути негативний (див. блок-схему в п. 1.1, рис. 1.2). Наступна програма ілюструє використання умовного оператора при визначенні дійсних коренів *x1, x2* квадратного рівняння:

```
        #include<stdio.h>
#include<math.h>
main()
{
        double a=6.7,b=1.2,c=4.3,x1,x2,d;
        d=b*b – 4*a*c; /* d – дискримінант    */
if  (d>=0.0)
{
  x1=(-b+sqrt(d))/2/a;
  x2=(-b-sqrt(d))/2/a;
  printf(«\n Корені:  xl=%e,   x2=%e»,  x1,  x2);
}
else
  printf(«\n дійсні корені відсутні.»);
        getchar();
}
```

В умовному операторі після *if* знаходиться складовою оператор, укладений у фігурні дужки, після *else* – тільки один оператор – виклик функції *printf ().*

У (*вираз_умова*) можуть використовуватися такі операції відносин (порівняння):

< менше, ніж (ранг 6); ,

> більше, ніж (ранг 6);

<= менше або дорівнює (ранг 6);

> = більше або дорівнює (ранг б);

== одно (ранг 7);

! = не дорівнює (ранг 7).

Операнди операцій відносин повинні бути арифметичного типу або можуть бути покажчиками. Результат операції цілочисловий: 0 (брехня) або 1 (істина). Останні дві операції (операції порівняння на рівність) мають нижчий ранг (пріоритет) в порівнянні з іншими операціями відносин. Таким чином, вираз

(X <B == A <x) одержить значення 1,

у тому випадку, коли значення x знаходиться в інтервалі від A до B і A < B, або x, A, B рівні. (Спочатку обчислюються x < B і A < x, до результатів застосовується операція порівняння на рівність ==).

Також у (*вираз_умова*) можуть використовуватися логічні бінарні операції:

&& – кон'юнкція («І») арифметичних операндів або відносин (ранг 11. Цілочисловий результат 0 (брехня) або 1 (істина);

|| – диз'юнкція («АБО») арифметичних операндів або відносин (ранг 12). Цілочисловий результат 0 (брехня) або 1 (істина).

Приклади результатів відносин і логічних операцій:

3 < 5 дорівнює 1;

3 > 5 дорівнює 0;

3 = 5 дорівнює 0;

3! = 5 дорівнює 1;

3! = 5 || 3 = 5 дорівнює 1;

3 + 4 > 5 && 3 + 5 > 4 && 4 + 5 > 3 дорівнює 1.

2.3.3 Умовна тримісна операція

На відміну від унарних і бінарних операцій умовна тернарна (тримісна) операція (ранг 13) використовується з трьома операндами. У зображенні умовної операції застосовуються два символу «?» і «:» і три вирази-операнда:

вираз_1? Вираз_2: вираз _3;

Першим обчислюється значення *вираз _1*. Якщо воно істинне, тобто не дорівнює нулю, то обчислюється значення *вираз _2*, яке стає результатом. Якщо при обчисленні *вираз _1* вийде 0, то в якості результату береться значення *вираз _3*. Класичний приклад:

x <0? X: x;

Цей вислів повертає абсолютну величину (модуль) змінної *x*.

2.4 Лабораторна робота № 3. Програмування розгалуженого обчислювального процесу з різними логічними умовами. Формальна логіка у розв'язанні задач діагностики, лікування та профілактики захворювань

Мета роботи: вивчити реалізацію в мові розгалужених обчислювальних процесів. Навчитися писати програми, використовуючи оператори розгалуження if … else.

Завдання 1. Обчислити значення функції $y = f(x)$, де

$$x = \begin{cases} f_1(z), & \text{если } z < 0, \\ f_2(z), & \text{если } 0 \le z < 8, \\ f_3(z), & \text{если } z > 8, \end{cases}$$

$$z = \cos(c).$$

Значення функцій наведені в табл. 1.3.

Завдання 2

2.1 При діагностиці захворювання шлунково-кишкового тракту визначають кислотність середовища РН-метрії і використовують наступні критерії: РН < 7 – середовище кисле, РН=7 – середовище нейтральне, РН > 7 – лужне середовище. Скласти алгоритм вирішення логічної задачі та запрограмувати його.

2.2. Створюється інформаційно-довідкова система, яка на запитання користувача видає довідку про різні лікарські препарати, їх дозування, показання та протипоказання. Треба розробити структурну схему алгоритму фрагмента цієї системи, в якій видаються довідки про дозування серцевого препарату корглікону залежно від віку пацієнта, виходячи з таких критеріїв:

Вік (років)	Доза (мг)
до 2	не призначається
2–6	0,1–0,5
6–12	0,5–0,75
більше 12	0,75–1,0

2.3 Створюється інформаційно-довідкова система, яка дає змогу у відповідь на запит користувача видати довідку про різні фізіологічні показники. Треба розробити структурну схему алгоритму фрагмента цієї системи, де розраховувався б об’єм вмісту води (ОВВ) для дорослого залежно від ваги і статі пацієнта, виходячи з таких критеріїв:

Стать чоловіча	Стать жіноча
ОВВ = Вага × 0,8	ОВВ = Вага × 0,75

Звіт повинен містити:

1) мету роботи;

2) умову задачі;

3) блок-схему алгоритму розв'язання задачі;

4) програму;

5) розв'язання задачі (лістинг) на ЕОМ;

6) короткі висновки з роботи.

Контрольні питання

1. Як організовуються чисельні дії в операторі умови *if*?
2. Який формат запису має тернарний оператор умови?
3. Який оператор умови рекомендується використовувати для програмування меню?
4. Які логічні оператори та оператори відношень використовуються в мові Сі?

2.5 Стратегії отримання медичних знань. Організація циклів. Оператор *switch*, оператор *break*, оператор *goto*. Організація мультірозгалуження в програмі

2.5.1 Форми операторів циклу

У мові Сі рівноправно використовуються три різних оператори циклу, що позначаються відповідно службовими словами ***while, for, do.***

Цикл ***while*** (цикл із передумовою) має вигляд:

while (вираз_умова)

тіло_циклу;

В якості *вираз_умова* найчастіше використовується відношення або логічний вираз. Якщо воно істинне, тобто не дорівнює 0, то тіло циклу виконується до тих пір, поки *вираз_умова* не стане хибним.

Перевірка істинності вираження здійснюється до кожного виконання тіла циклу. Таким чином, для завідомо неправдивого *вираз_умова* тіло циклу не виконається жодного разу. *Вираз_умова* е може бути і арифметичним виразом. У цьому випадку цикл виконується, поки значення *вираз_умова* не дорівнює 0.

Цикл ***do*** (цикл з умовою поста) має вигляд:

do
тіло_ циклу
while (вираз_умова);

Вираз_умова логічне або арифметичне, як і в циклі ***while***. У циклі ***do*** тіло циклу завжди виконується принаймні один раз. Після кожного виконання тіла циклу перевіряється істинність *вираз_умова* (на рівність 0), і якщо воно помилкове (тобто дорівнює 0), то цикл закінчується. В іншому випадку тіло циклу виконується знову.

Цикл ***for*** (званий параметричним) має вигляд:

for (вираз_1; вираз_умова; вираз _3)
тіло_циклу;

Перше і третє вираження в операторі ***for*** можуть складатися з декількох виразів, розділених комами. *Вираз_1* визначає дії, що виконуються до початку циклу, тобто задає початкові умови для циклу; найчастіше це вираз присвоювання. *Вираз_умова* – зазвичай логічне або арифметичне. Воно визначає умови закінчення або продовження циклу. Якщо воно істинне (тобто не дорівнює 0), то виконується тіло циклу, а потім обчислюється

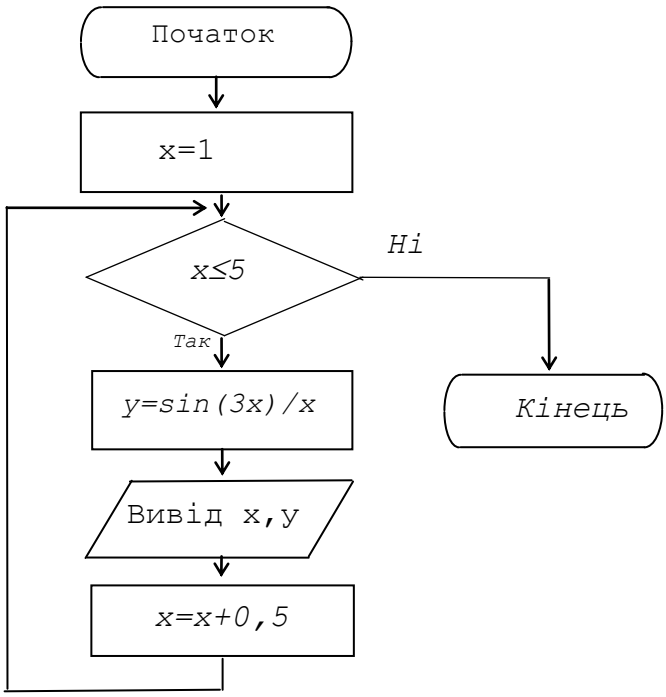
вираз _3. Вираз _3 зазвичай задає необхідні для наступної ітерації зміни параметрів або будь-яких змінних тіла циклу. Після виконання *вираз* _3 обчислюється істинність *вираз_умова*, і все повторюється. Таким чином, *вираз_1* обчислюється тільки один раз, а *вираз_умова* і *вираз_3* обчислюються після кожного виконання тіла циклу.

Цикл продовжується до тих пір, поки не стане хибним *вираз_умова*. Будь-яке з трьох, будь-які два або всі три вирази в операторі ***for*** можуть бути відсутніми, але розділяють їх символи «;» повинні бути присутніми завжди. Якщо відсутня *вираз_умова*, то вважається, що воно істинне і потрібні спеціальні засоби для виходу з циклу.

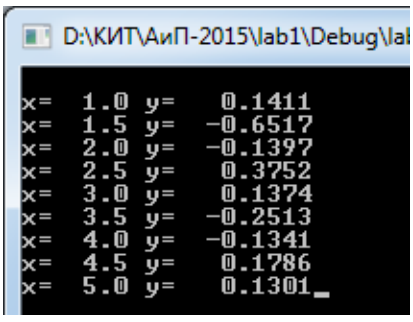
Проілюструємо особливості трьох типів циклу на наступному прикладі.

Приклад 6. Обчислити $y = sin(3x)/x$, якщо $1 \le x \le 5$ з кроком 0,5.

2) цикл з передумовою (рисунки 2.2. *а)*, 2.2.*б)*)



а)



б)

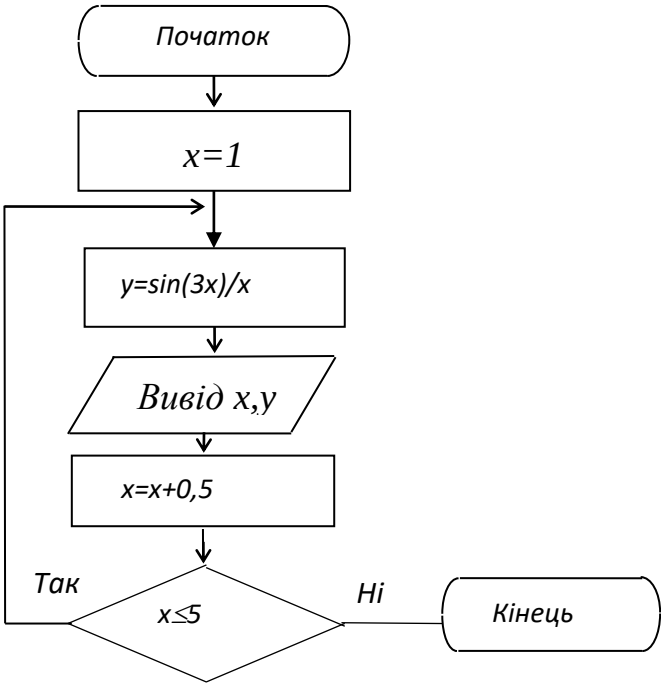
а – блок-схема алгоритму;
б – результат розв'язання циклу з передумовою

Рисунок 2.2

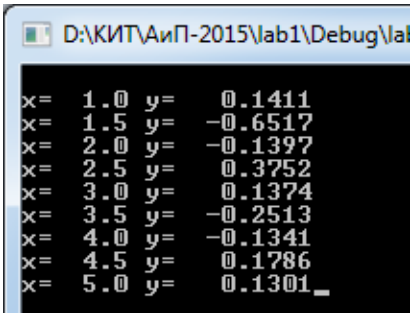
```
#include<stdio.h>
#include<math.h>
main()
{
        double x,y;
        x=1;
        while (x<=5)
        {
                y=sin(3*x)/x;
                printf(«\nx=%5.1f\ty=%9.4f»,x,y);

                x=x+0.5;
        }
        getchar();
}
```

2) цикл с постумовою (рисунки 2.3, *а), б)*)



а)



б)

а – блок-схема алгоритму
б – результат розв'язання циклу з передумовою

Рисунок 2.3

```
#include<stdio.h>
#include<math.h>
main()
{
        double x,y;
        x=1;
        do
        {
                y=sin(3*x)/x;
                printf("\nx=%5.1f\ty=%9.4f",x,y);
                x=x+0.5;
        }
        while (x<=5);
        getchar();
}
```

3) Цикл параметричний

```
#include<stdio.h>
#include<math.h>
main()
{
        double x,y;
                for(x=1;x<=5;x=x+0.5)
        {
                y=sin(3*x)/x;
                printf("\nx=%5.1f\ty=%9.4f",x,y);
        }
        getchar();
}
```

2.5 Використання інкременту й декременту у мові Сі

У мові Сі визначені два оператора, які здійснюють збільшення або зменшення цілочисленої величини на 1:

• оператор ++ – інкремент;

• оператор - - декремент.

Ці оператори є унарними. Вони вимагають одного операнда. Ці оператори можуть розміщуватися до і після операнду.

Оператор інкременту ++ збільшує значення операнду на 1. Наприклад, рядок

x = x + 1;

є аналогічна рядку

x ++;

або

++ x;

Так само, оператор декременту зменшує значення операнду на 1. Наприклад, рядок *x = x – 1;* можна записати

x -;

або

- x;

Приклад 7

```
// оператори інкременту (++) та декременту (--)

int a, b;

a = 10;
b = a++; // b = 10; a = 11

a = 10;
b = ++a; // b = 11; a = 11

a = 10;
b = a--; // b = 10; a = 9

a = 10;
b = --a; // b = 9; a = 9
```

2.6 Лабораторна робота № 4. Організація циклів

Мета роботи. Вивчити написання програм на мові Сі, використовуючи ітераційні (циклічні) методи, освоїти основні оператори, що підтримують роботу з циклами (***for, while, do ... while***). Навчитися писати програми, використовуючи дані оператори.

Лабораторна робота містить 3 завдання.

Завдання 1. Визначте таблицю значень функції

$$y = \begin{cases} f_1(x), & \text{якщо} \quad x \le a, \\ f_2(x), & \text{якщо} \quad x > a \end{cases}$$

для значень аргументу *x* в інтервалі від *Xn* до *Xк* з кроком *ΔX*. Вихідні дані наведені в таблиці 2.5.

Таблиця 2.5 – Вихідні дані

Вар.	$f_1(x)$	$f_2(x)$	***a***	***Xн***	***Xк***	***ΔX***
1	2	3	4	5	6	7
1	$\frac{a+e^x}{\cos x+x^2+a}$	$ax+\sqrt{\lvert x\rvert}$	π/4	-π	π	π/3
2	$\frac{a\ln^2 x}{a+\sqrt{x}}$	$\frac{\sin^2 x}{\sqrt{x}+ax}$	4,1	1,2	3,6	0,2
3	$\frac{\ln^2(x+a)}{a\sqrt{x}}$	$\frac{e^{ax}+5}{1+\cos^2 x}$	2,8	1,4	4,2	0,3
4	$\frac{\ln^3(a+x)}{a+\sqrt{x}}$	$e^{ax}+\cos^2 x$	3,9	2,8	4,6	0,2
5	$\frac{\sqrt{ax}-a}{tg^3 a}$	$\frac{e^{-x}(1+ax)}{\ln^2 x}$	2,4	0,7	3,8	0,2
6	$\frac{\cos^2 x-a}{a\sqrt{x}}$	$\frac{e^{-ax}}{\ln(x+a)}$	π	0,1	2π	π/6
7	$\frac{x\sin\sqrt{x}}{\pi a^2+8}$	ax^2+e^x	2,1	0,8	3,6	0,2
8	$\frac{\sin^3(ax)}{ax+5}$	$\frac{e^{-(a+x)}}{a+\cos^3(ax)}$	5,4	2,3	8,9	0,4

Продовження таблиці 2.5

1	2	3	4	5	6	7
9	$\frac{\lvert ax+8\rvert}{ln\lvert x\rvert}+cos^2 x$	$\sqrt{1+e^{ax}}$	-1	-4,2	3,8	0,5
10	$\frac{1+\sqrt{ax}}{0{,}5+sin^2(ax)}$	$\frac{a-e^{a-x}}{ln^2 x}$	14,2	11,6	1,6	0,2
11	$\frac{(a+x)^2}{2+cos^3(ax)}$	$\frac{a+sin^2(ax)}{e^{-x/2}}$	1,1	0,2	1,8	0,2
12	$\frac{sin^2 x-a}{ax}$	$\frac{atg^2 x}{a+0{,}7x}$	5,4	2,2	7,3	0,3
13	$\frac{ln(a^2+x)}{asin^2 x}$	$\frac{a-\sqrt{ax}}{1+cos^2 x}$	2,5	1,9	3,8	0,2
14	$\frac{ln^2(x+a)}{(a+x)^3}$	$\frac{\sqrt{alnx}}{1+tg^2(ax)}$	5,1	3,3	6,9	0,3
15	$\frac{1+tg^2(x/a)}{a+e^{x/a}}$	$\frac{e^{ax}+a^x}{\sqrt{1+e^{ax}}}$	0,7	0,6	0,9	0,05
16	$\frac{\sqrt{ax+5}}{ln^2 x}$	$\frac{1+sin^2(a+x^2)}{\sqrt[3]{a+x^2}}$	3,8	1,2	5,3	0,4
17	$\frac{a}{x^5(e^{ax}+1)}$	$\frac{10x-e^{ax}}{ax+2}$	2,7	1,8	4,2	0,3
18	$\frac{sin(\pi ax)}{ln(a+x^2)}$	$ax+\sqrt{\frac{x}{a}}$	2,1	0,4	3,9	0,2
19	$\frac{5a+tgx^2}{\sqrt{\lvert lnx\rvert}}$	$\frac{\sqrt{\lvert sin(ax)\rvert}}{1+x^3}$	3,4	0,5	4,6	0,5
20	$\frac{ln\lvert tg(ax)\rvert}{1+ax^2}$	$asin(x^3)+x$	2,8	1,6	4,8	0,3
21	$\frac{ln(x+0{,}7)}{4x^2+ax}$	$x^4 sin(ax)$	4,2	2,8	5,6	0,2
22	$\frac{asin^2 x}{e^{ax}+8}$	$ax+tg(ax)$	2,6	0,5	3,5	0,5
23	$\frac{alnx}{x}+x^4$	$\sqrt{ax^2+sinx}$	6,3	2,5	7,5	0,5
24	$\frac{e^{ax}}{\sqrt{x^2+\pi a^2}}$	$e^{\sqrt{x}}(1+ae^x)$	5	1	10	0,5
25	$\frac{ae^x sin^3 x}{ax+1}$	$a\sqrt{e^x+\pi}$	π	0	2π	π/4

Завдання 2. Знайти суму ряду $y = \sum \frac{f_1}{f_2}$, де $a \le x \le b$, $\Delta x = c$.

Варіанти завдань наведені в таблиці 1.5.

Завдання 3.

Табулювання функцій.

Скласти таблицю значень тиску крові в аорті $P = P\,0\,e^x$ у діапазоні 0 < t < 1 (с) з кроком At = 0,1 (с). P0 – початкове значення тиску крові, x – гідравлічний опір аорти, k – еластичність аорти.

Звіт повинен містити:

1) мету роботи;

2) умову задачі;

3) блок-схему алгоритму розв'язання задачі;

4) програму;

5) рішення задачі (лістинг) на ЕОМ;

6) короткі висновки з роботи.

Контрольні питання

1. Як організовуються складові оператори циклів на мові **Сі**?
2. Як організовуються вкладені цикли на мові **Сі**?
3. В яких випадках може статися зациклення при використанні оператора циклу з передумовою?
4. В яких випадках може статися зациклення при використанні оператора циклу з постумовою?
5. Як реалізується взаємозамінність операторів циклу while і for?
6. У чому подібність і відмінність між циклами з передумовою та з постумовою?

2.7 Оператор switch, оператор break, оператор goto

2.7.1 Організація мультірозгалуження в програмі

Основним засобом для організації мультірозгалуження служить оператор-перемикач, формат якого має вигляд :

switch (вираз)
{case константа1: оператори_1;
case константа2: оператори_2;
.....
default: оператори;
}

У цьому операторі використовуються три службових слова: **switch, case, default.** Перше з них ідентифікує власне оператор-перемикач. Службове слово **case** з подальшою константою є в деякому сенсі міткою. Константи можуть бути цілими або символьними і всі повинні бути різними (щоб мітки були помітні). Службове слово **default** також позначає окрему мітку. При виконанні оператора обчислюється вираз, записаний після **switch**, і його значення послідовно порівнюється з константами, які поміщені слідом за **case**. При першому ж збігу виконуються оператори, помічені даної міткою. Якщо виконані оператори не передбачають будь-якого переходу (тобто серед них немає ні **goto**, ні **return**, ні **exit**(), ні **break**), то далі виконуються оператори всіх наступних варіантів, поки не з'явиться оператор переходу або не закінчиться перемикач.

Оператори слідом за **default** виконуються, якщо значення виразу в дужках після **switch** не співпало з жодною константою після **case.** Мітка **default** може в перемикачі відсутні. В цьому випадку при розбіжності значення виразу з константами перемикач не виконує ніяких дій. Оператори, помічені міткою **default**, не обов'язково знаходяться в кінці (після інших варіантів перемикача). Уточнимо, що **default** і "**case** константа" не є мітка-

ми в звичайному сенсі. До них, наприклад, не можна перейти за допомогою оператора **goto**.

Оператор **break**. Цей оператор припиняє виконання оператора циклу і передає управління наступному за ним (за циклом) оператору.

Необхідність у використанні оператора переривання в тілі циклу виникає, коли умову продовження ітерацій потрібно перевіряти не спочатку циклу (як у циклах **for** і **while)** і не в кінці тіла циклу (як в циклі **do**), а в середині тіла циклу. Найбільш природна в цьому випадку така структура тіла циклу:

{
операторі
if *(умова)* break;
операторі
}

Наприклад:

```
while(1)
{
printf ("\n Ввести значення n=”);
scanf("%d",&n);
if( n > 0 ) break;
printf ("\n помилка! n треба >0!\n”);
}
```

Оператор continue. Ще одну можливість впливати на виконання операторів тіла циклу забезпечує оператор переходу до наступної ітерації циклу **continue.** Оператор **continue** протилежний за дією оператору **break**. Він дозволяє в будь-якій точці тіла циклу перервати поточну ітерацію і перейти до перевірки умов продовження циклу, визначених у пропозиціях **for** або **while.** Відповідно до результатів перевірки виконання циклу або закінчується, або починається нова ітерація. Оператор **continue** зручний,

коли від ітерації до ітерації змінюється послідовність виконуваних операторів тіла циклу, тобто коли тіло циклу містить розгалуження. Розглянемо на наступному прикладі.

Приклад 8. Ввести послідовність довільних дійсних чисел, що закінчується нулем. Обчислити суму і кількість тільки позитивних чисел.

```
#include <stdio.h>
/* Сума позитивних чисел  */
void main( )
{
double s,x;/*x - чергове число, s - сума*/
int k; /*k - кількість позитивних */
printf("\n Ввести послідовність чисел"
" с 0 наприкінці:\n");
for( x=1.0, s=0.0, k=0; x !> 0.0; )
{
scanf("%lf",&x);
if( x <= 0.0 ) continue;
k++;s+=x;
}
printf("\n Сума=%f, кількість позитивних =%<d" ,s,k) ;
}
```

Результат виконання програми:

Введіть послідовність чисел з 0 у кінці:

-3.0 14.0 -5 -4 10 0.0

Сума = 30.000000, кількість позитивних = 3

Недолік наведеної програми полягає в тому, що немає захисту від невірно введених даних. Наприклад, не передбачені дії, коли в послідовності відсутній нульовий елемент.

Приклад 9. Інструкція застосування препарату Корглікон (табл. 2.6)

Таблиця 2.6 – Інструкція застосування препарату Корглікон

Вік (років)	**Доза (мг)**
До 2	не призначається
2–6	0,1–0,5
6–12	0,5–0,75
більше 12	0,75–1,0

Ураховуючи правила складання структурної схеми, починаємо створення схеми з блоку "Початок". Оскільки дозування препарату залежить від віку пацієнта, то в наступних блоках визначимо вік пацієнта та перевіримо його належність до кожного з вікових проміжків, визначених в інструкції (блок перевірки тверджень – ромб). Від блоку перевірки тверджень відходить завжди дві стрілки – "так" і "ні". Над стрілкою "так" дамо вказівку (блок у вигляді прямокутника) про дозування препарату відповідно до певного вікового проміжку. За стрілкою "ні" продовжимо перевірку наступного вікового періоду (рисунок 2.4).

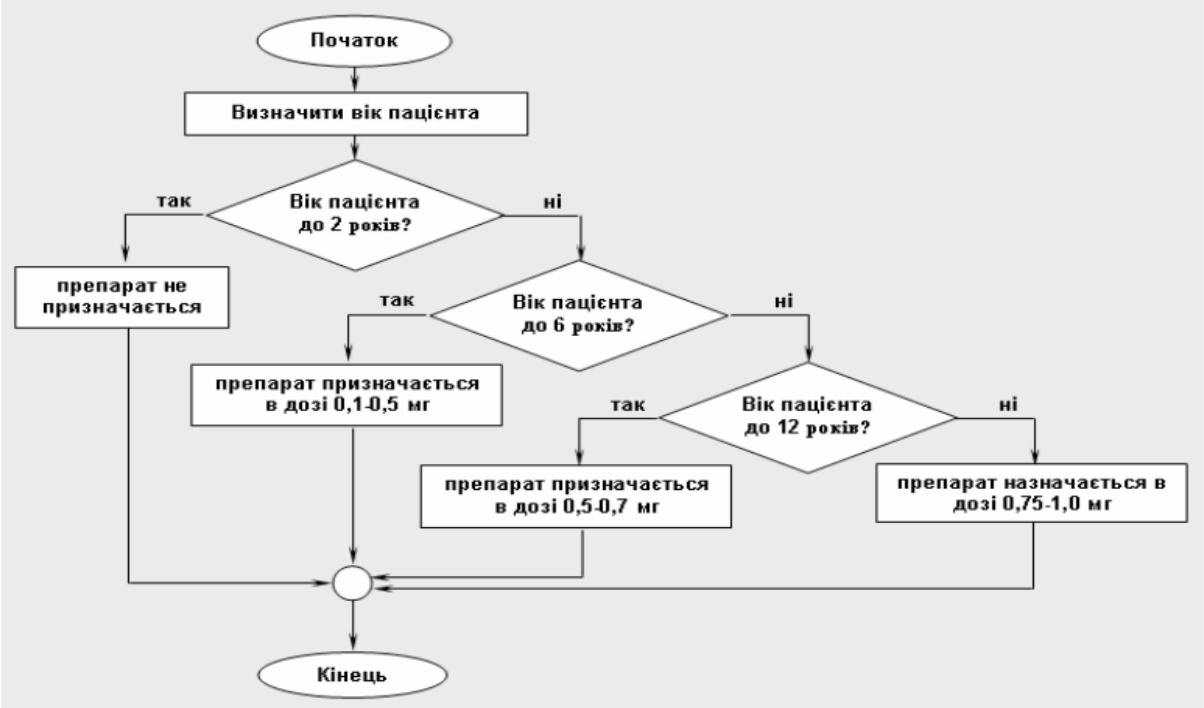


Рисунок 2.4. Структурна схема алгоритму дозування серцевого препарату Корглікону

2.8 Лабораторна робота № 5. Організація мультірозгалуження в програмі. Клінічні системи підтримки прийняття рішень

Мета роботи: навчитися писати програми, використовуючи оператори перемикання **switch** в парі з оператором **break**, оператор переходу **goto.**

Завдання 1. Використовуючи оператор s**witch**, складіть програму для вирішення наступного завдання.

Варіант mn. Обчисліть таблицю значень функції

$$y = \begin{cases} f_1(x), & якщо \quad x \in X_1, \\ f_2(x), & якщо \quad x \in X_2, \\ f_3(x), & якщо \quad x \in X_3, \\ f_4(x), & якщо \quad x \in X_4 \end{cases}$$

для цілочислових значень аргументу *x* з інтервалу [*Xn, Xк*]. Множини *X1, X2, X3, X4* і функції *f1, f2, f3, f4* задані в таблицях 2.7, 2.8.

Таблиця 2.7 – Множини X1, X2, X3, X4

m	***Xн***	***Xк***	***X₁***	***X₂***	***X₃***	***X₄***
0	-10	15	непарні числа з інтервалу [6, 12]	парні числа з інтервалу [6, 12]	[-2, 5]	інші числа
1	0	30	парні числа з інтервалу [10, 20]	непарні числа з інтервалу [10, 20]	[1, 8]	інші числа
2	1	25	числа, кратні 3 з інтервалу [10, 20]	числа, некратні 3 з інтервалу [10, 20]	[2, 5] та [21, 24]	інші числа

Таблиця 2.8 – Функції f1, f2, f3, f4

n	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
1	$\pi x^2+\sin x$	$10x^3-tg(x/5)$	$\cos \pi x$	$e^{-x/4}$
2	$\ln^2(x^2+1,5)$	e^{-2x}	$\sin \pi x$	$arctg(3x)$
3	e^{2x-5}	$\cos^2(\pi x)$	$\ln(x^2+2,5)$	$arcsin(x/30$
4	$arccos(x/35)$	e^{2x-10}	$\ln^2(3x+1)$	$\sin(2\pi x)$
5	$e^{-x/10}$	$arctg(4x)$	$\sqrt{x^2+8}$	$\lvert 8x^3-20\rvert$
6	$\sqrt{\lvert x-10\rvert}$	$tg^2(x/5)$	$2\sin(x/\pi)$	e^{x+2}
7	e^{2x-15}	$\sqrt{5x^2+1}$	$\ln\lvert x+1,5\rvert$	$\sin^3(x^2)$
8	$5e^{-x}$	$\sin(\pi x^2/2)$	$\sqrt{\lvert \ln x^2\rvert}$	$arctg(x/3)$
9	$\pi \sin(\pi x/3)$	$e^{x/10+2}$	$\sqrt{x^3+4}$	$\cos^2\lvert x\rvert$
0	$\sqrt{1+\cos^2(2x)}$	$\ln(x^3+1,8)$	$\sin(3x+\pi)$	x^2+5x

Завдання 2. Клінічні системи підтримки прийняття рішень

2.1 Скласти структурну схему алгоритму, програму, яка виводить на екран дозування препарату «Імунал» залежно від лікарської форми та віку пацієнта згідно з формалізованою інструкціїю застосування (табл. 2.9).

Таблиця 2.9 – Інструкція застосування препарату «Імунал»

Вік (років)	**Краплі**	**Таблетки**
Діти від 1 до 6 років	1 мл 3 рази на день	З 4 років 1 табл. 1–3 р на день
Діти від 6 до 12 років	1,5 мл 3 рази на день	1 табл. 1–3 рази на день
Дорослі й діти старше 12 р.	2,5 мл 3 рази на день	1 табл. 3–4 рази на день

2.2 Скласти структурну схему алгоритму, програму, яка виводить на екран визначення виду кровотечі за певними ознаками.

Розрізняють кровотечу артеріальну, венозну та капілярну. Найбільш небезпечна артеріальна кровотеча, ознаками якої є витік крові пульсуючим потоком у вигляді фонтану яскраво-червоного кольору. Венозна кровотеча виникає при ушкодженні вен. Тиск у венах значно менший за артеріальний, тому кров витікає повільно, рівномірно-неперервним потоком темно-вишневого кольору. Капілярна кровотеча є наслідком ушкоджень дрібних кровоносних судин (капілярів) і характеризується тим, що з усієї поверхні

рани сочиться кров у невеликій кількості, така кровотеча у більшості випадків через деякий час зупиняється самостійно внаслідок природного зсідання крові.

2.3 У лабораторії медико-біологічних досліджень треба підтримувати температуру, яка дорівнює 26 "С. У розпорядженні дослідника є: нагрівник, кондиціонер, вимірювач температури, ЕОМ. Розробити структурну схему алгоритму керування ЕОМ всією зазначеною апаратурою для отримання необхідного результату.

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв'язання задачі; програму;

3) рішення задачі (лістинг) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. У чому відмінність і подібність між операторами **break** і **continue**?

2. Що станеться, якщо в операторі **switch** після мітки **case** не використовувати оператор **break**?

3. Що станеться, якщо в операторі **switch** не поставити мітку **default** і умова перемикання не збігається ні з однією міткою **case**

3 ОДНОВИМІРНІ ЧИСЛОВІ МАСИВИ. СЕЛЕКТИВНЕ ОБРОБЛЕННЯ ЕЛЕМЕНТІВ МАСИВУ МЕДИЧНИХ ДАНИХ. ЗНАХОДЖЕННЯ МІНІМАЛЬНОГО Й МАКСИМАЛЬНОГО ЕЛЕМЕНТІВ МАСИВУ. СОРТУВАННЯ ОДНОВИМІРНИХ МАСИВІВ МЕДИЧНИХ ДАНИХ

3.1 Одновимірні масиви

Математичним поняттям, яке призвело до появи в мовах програмування поняття «масив», є матриця і її окремі випадки: вектор-стовпець або вектор-рядок. Елементи матриць в математиці прийнято позначати з використанням індексів. Істотно, що всі елементи матриць або речові, або цілі і тощо. Така «однорідність» елементів властива і масиву, визначення якого описує тип елементів, ім'я масиву і його розмірність, тобто число індексів, необхідне для позначення конкретного елемента. Крім того, в ухвалі зазначається кількість значень, прийнятих кожним індексом.

Одновимірний масив – це список пов'язаних однотипних змінних.

Загальна форма запису одновимірного масиву:

тип ім'я_масива[розмір];

У наведеному записі елемент *тип* оголошує базовий тип масиву. Кількість елементів, які будуть зберігатися в масиві з ім'ям *ім'я_масива*, визначається елементом *розмір*.

Доступ до окремого елемента масиву здійснюється за допомогою індексу. Індекс описує позицію елемента всередині масиву.

Усі масиви займають суміжні комірки пам'яті, тобто елементи масиву в пам'яті розташовані послідовно один за одним. Комірка пам'яті з найменшою адресою належить першому елементу масиву, а з найбільшою – останньому.

Наприклад,

int a [10]; визначає масив з 10 елементів a [0], a [1], ..., a [9].

float Z [13] [6]; визначає двовимірний масив, перший індекс якого приймає 13 значень від 0 до 12, другий індекс приймає 6 значень від 0 до 5. Таким чином, елементи двовимірного масиву Z можна перерахувати так:

Z [0] [0], Z [0] [1], Z [0] [2], ..., Z [12] [4], Z [12] [5]

відповідно до синтаксису Сі в мові існують тільки одномірні масиви, однак елементами одновимірного масиву, у свою чергу, можуть бути масиви. Тому двовимірний масив визначається як масив масивів. Таким чином, в прикладі визначений масив Z із 13 елементів-масивів, кожен з яких, у свою чергу, складається з 6 елементів типу *float.* Нумерація елементів будь-якого масиву завжди починається з 0, тобто індекс змінюється від 0 до N-1, де N – кількість значень індексу.

Обмежень на розмірність масивів, тобто на число індексів у його елементів, в мові Сі теоретично немає. Стандарт мови Сі вимагає, щоб транслятор міг обробляти визначення масивів з розмірністю до 31. Однак найчастіше використовуються одномірні і двовимірні масиви. Розглянемо на простих обчислювальних задачах деякі прийоми роботи з масивами.

Приклад 10. Обчислення середнього. Ввівши значення n з діапазону (0 <n <= 100) і значення n перших елементів масиву x [0], x [1], ..., x [n-1], обчислити середнє значень введених елементів масиву. Завдання вирішує така програма:

```
/* Обчислення середнього */
#include <stdio.h>
void main ( )
{      /*n – кількість елементів */
int i,n; /*b-середнє*/
float b,x[100];
while (1)
```

```
{
printf("\n Ввести значення n=");
scanf("%d", &n);
if ( n > 0 && n <= 100 ) break;
printf("\n Помилка! Необхідно 0<n<101 ") ;
} /* Кінець циклу вводу значення n */
printf("\n Ввести значення елементів:\n");
for( b=0.0,i=0; i<n; i++)
{
printf("x[%d] =", i);
scanf("%f", &x[i]);
b+=x[i];      /* Обчислення суми  елементів */
}
b/=n;/* Обчислення середнього */
printf("\n Середнє =%f, b) ;
}
```

У програмі визначено масив *X* зі 100 елементами, хоча в кожному конкретному випадку використовуються тільки перші n з них. Введення значення n супроводжується перевіркою допустимості введеного значення. Як умова після **while** записано свідомо справжній вираз 1, тому вихід з циклу (оператор **break**) можливий тільки при введенні правильного значення *n*, що задовольняє нерівності 0 < n < 101. Наступний цикл забезпечує введення *n* елементів масиву й отримання їхньої суми (*b*).

3.1.1 Ініціалізація масивів

Ініціалізація – це об'єднання визначення об'єкта з одночасним привласненням йому конкретного значення. Використання ініціалізації дозволяє змінити формат визначення масиву. Наприклад, можна явно не вказувати кількість елементів одновимірного масиву, а тільки перерахувати їх початкові значення в списку ініціалізації:

double d[ ]={1.0, 2.0, 3.0, 4.0, 5.0};

У цьому прикладі довжину масиву компілятор обчислює за кількістю початкових значень, перерахованих в фігурних дужках. Після такого визначення елемент d [0] дорівнює 1.0, d [l] дорівнює 2.0 і т. д. до d [4], який дорівнює 5.0.

Якщо у визначенні масиву явно вказано його розмір, то кількість початкових значень не може бути більшою за кількість елементів в масиві. Якщо кількість початкових значень менше, ніж оголошена довжина масиву, то початкові значення отримають тільки перші елементи масиву (з меншими значеннями індексу):

int M[8]={8,4,2};

У цьому прикладі визначені значення тільки змінних M [0], M [1] і M [2], рівні відповідно 8, 4 і 2. Елементи M [3], ..., M [7] не ініціалізуються.

3.1.2 Селективне оброблення елементів масиву

Селективне оброблення полягає в обробленні тільки тієї інформації, яка задовольняє заданій умові.

Найбільш часто зустрічаються такі умови оброблення елементів масиву:

- *парні* — *A[i] %2 ==0*
- *непарні* — *A[i] %2 !=0*
- *кратні k* — *A[i] % k ==0*
- *не кратні k* — *A[i] % k !=0*
- *стоять на парних місцях* — *(i+1) % 2==0*
- *стоять на непарних місцях* — *(i+1) % 2!=0*
- *додатні* — *A[i] >0*
- *від'ємні* — *A[i] <0*
- *в інтервалі (x1,x2)* — *((A[i]>x1) && (A[i]<x2))*

Приклад 11. Підрахувати кількість позитивних елементів у масиві *x(10).*

Порядок роботи:

Крок 1. Уводимо масив *x(10).*

Крок 2. Задаємо початкове значення кількості $k = 0$.

Крок 3. Організовуємо цикл, що перебирає елементи масиву (тобто індекс *i*), починаючи з 0-го і закінчуючи 9-м.

Крок 4. Якщо $x_i > 0$, тоді привласнюємо $k = k + 1$.

Крок 5. Якщо цикл за *i* не закінчився ($i < 10$), йдемо на початок циклу, тобто на крок 3, інакше крок 6.

Крок 6. Друкуємо *k.*

Крок 7. Кінець алгоритму.

3.2 Лабораторна робота № 6. Селективне оброблення масивів. Селективне оброблення елементів масиву медичних даних

Мета роботи: вивчити роботу з масивом як зі складеним типом даних, прийоми введення і виведення даних, оброблення одновимірних масивів; виробити практичні навички складання алгоритмів, написання й налагодження програм у задачах селекції в одновимірному масиві.

Завдання 1. Скласти блок-схему і програму для оброблення масиву відповідно до індивідуального завданням, наведеного в таблиці 3.1. Розмір і ім'я масиву вибрати самостійно.

Таблиця 3.1 – Індивідуальні завдання

Вар.	**Умова задачі**
1	Знайти суму парних чисел масиву
2	Обчислити добуток негативних чисел масиву
3	Визначити кількість непарних чисел масиву
4	Знайти суму негативних чисел масиву
5	Визначити кількість негативних чисел масиву
6	обчислити добуток позитивних чисел масиву
7	Знайти суму позитивних чисел масиву
8	Визначити кількість парних чисел масиву
9	обчислити добуток парних чисел масиву
10	Знайти суму непарних чисел масиву
11	визначити кількість кратних 3 чисел масиву
12	Обчислити добуток непарних чисел масиву
13	Знайти суму кратних 3 чисел масиву
14	визначити кількість не кратних 3 чисел масиву
15	Обчислити добуток кратних 3 чисел масиву
16	Знайти суму не кратних 3 чисел масиву
17	Визначити кількість кратних 4 чисел масиву
18	Обчислити добуток не кратних 3 чисел масиву
19	Знайти суму кратних 4 чисел масиву
20	Обчислити добуток кратних 4 чисел масиву
21	Знайти суму не кратних 4 чисел масиву
22	Визначити кількість кратних 5 чисел масиву
23	Обчислити добуток не кратних 4 чисел масиву
24	Знайти суму кратних 5 чисел масиву
25	Знайти середнє геометричне негативних непарних елементів масиву
26	Знайти добуток негативних не кратних п'яти елементів масиву
27	Знайти середнє арифметичне елементів масиву, що знаходяться в інтервалі [- 10, 20]
28	Знайти середнє геометричне елементів масиву, що знаходяться в інтервалі [5, 20]
29	Обчислити середнє арифметичне додатних парних елементів масиву
30	Визначити кількість не кратних 4 чисел масиву

Завдання 2. Скласти блок-схему та програму, яка виводить на екран середнє арифметичне значення показників цін на таблетки в 10–15 аптеках міста (таблиця 3.2). Дані масиву ввести з клавіатури.

Таблиця 3.2 – Список препаратів

№ вар	***Препарати (інформацію про ціни на препарати взяти в Інтернеті)***
1	Парацетамол-Дарниця таблетки 500 мг
2	Аскофен-Л таблетки №10
3	Ризоптан таблетки 10 мг
4	Темпалгин таблетки знеболюючі N10
5	Супрадин Ведмежуйки пастилки жув. № 30
6	Супервіт таблетки жув. № 30
7	АлфаВіт Школяр таблетки жув. № 60
8	Доппельгерц Kinder Мультивітамінний комплекс пастилки жувальні №60
9	Анантаваті таблетки, в/плів. обол. № 30
10	Гліцин Форте Евалар таблетки по 0,3 г № 20
11	Барбовал краплі ор. по 25 мл
12	Нурофєн форте таблетки, в/о по 400 мг № 12
13	Ессенціалє форте Н капсули по 300 мг № 30
14	Гліцисед Макс таблетки № 30
15	Артифлекс крем по 40 г у тубах
16	Алохол таблетки, в/плів. обол. № 50
17	Ібупрофен таблетки, в/о по 200 мг № 50
18	Корвалтаб таблетки №20
19	Пірацетам-Дарниця таблетки по 400 мг № 30
20	Нейроксон таблетки, в/плів. обол. по 500 мг № 20
21	Холосас сироп по 250 г
22	Кардіомагніл таблетки, в/плів. обол. по 75 мг № 100
23	Магнікор таблетки, в/плів. обол., форте №100
24	Гепабене капсули тв. № 30
25	Когнум таблетки по 250 мг № 50

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв'язання задачі; програму;

3) рішення задачі (лістинг) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. Як організовуються одномірні числові масиви у мові Сі?
2. Як організується індексування числових масивів у мові Сі?
3. На кого або на що покладається контроль меж числових масивів у мові програмування Сі?
4. Для чого застосовується початкова ініціалізація числових масивів при подальшому їх використанні?
5. Умови селективного оброблення елементів масиву.

3.3 Знаходження екстремального значення

Наступним досить поширеним алгоритмом оброблення одновимірного масиву є пошук екстремального (мінімального або максимального) елемента масиву. Розглянемо на прикладах.

Приклад 12. Знайти мінімальний елемент з інтервалу [5, 12] у масиві *x*(15).

Порядок роботи:

Крок 1. Вводимо масив *X(15).*

Крок 2. Задаємо початкове значення мінімального елемента $Xmin = 10^{20}$.

Крок 3. Організовуємо цикл, що перебирає елементи масиву (тобто індекс *i*), починаючи з 0-го і закінчуючи 14-м.

Крок 4. Якщо x_i не належить інтервалу [5, 12], тоді йдемо на крок 6.

Крок 5. Якщо $x_i < Xmin$, тоді привласнюємо $Xmin = x_i$.

Крок 6. Якщо цикл по і не закінчився ($i < 15$), йдемо на початок циклу, тобто на крок 3, інакше – на крок 7.

Крок 7. Друкуємо *Xmin*.

Крок 8. Кінець алгоритму.

Приклад 13. Знайти максимальний елемент і його номер у масиві *X(30)*. Порядок роботи:

Крок 1. Вводимо масив *X(30)*.

Крок 2. Задаємо початкові значення максимального елемента і його номера: $X_{max} = X[0]$, $n_{max} = 0$.

Крок 3. Організовуємо цикл, що перебирає елементи масиву (тобто індекс *i*), починаючи з 1-го і закінчуючи 29-м.

Крок 4. Якщо $x_i > X_{max}$, тоді привласнюємо: $X_{max} = X[i]$, $n_{max} = i$.

Крок 5. Якщо цикл за *i* не закінчився ($i < 30$), йдемо на початок циклу, тобто на крок 3, інакше – на крок 6.

Крок 6. Друкуємо X_{max}, n_{max}.

Крок 7. Кінець алгоритму.

Блок-схема алгоритму приведена на рисунку 3.1

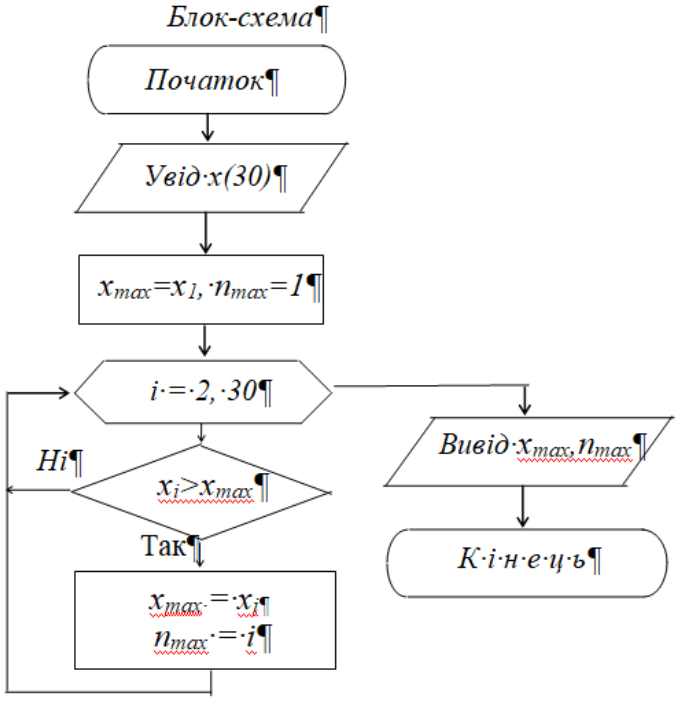


Рисунок 3.1 – Блок-схема алгоритму прикладу 13.

3.4 Лабораторна робота № 7. Знаходження мінімального та максимального елементів масиву

Мета роботи: вивчити роботу з масивом як зі складеним типом даних, прийоми введення і виведення даних, оброблення одновимірних масивів; виробити практичні навички в складанні алгоритмів, написанні і налагодженні програм в задачах пошуку екстремуму в одновимірному масиві.

Завдання 1. Скласти блок-схему і програму для оброблення масиву відповідно до індивідуального завданням, наведеного в таблиці 3.3. Розмір й ім'я масиву вибрати самостійно.

Таблиця 3.3 – Індивідуальні завдання

Вар.	**Умови задачі**
1	2
1	Знайти значення мінімального парного числа масиву
2	Знайти значення максимального парного числа масиву
3	Знайти значення мінімального непарного числа масиву
4	Знайти значення максимального непарного числа масиву
5	Знайти значення мінімального позитивного числа масиву
6	Знайти значення максимального негативного числа масиву
7	Знайти значення мінімального парного числа масиву та його номер
8	Знайти значення максимального парного числа масиву та його номер
9	Знайти номер і значення найменшого позитивного непарного елемента масиву
10	Знайти значення максимального непарного числа масиву та його номер
11	Знайти значення мінімального позитивного числа масиву та його номер
12	Знайти значення максимального негативного числа масиву та його номер
13	знайти значення мінімального кратного 3 числа масиву
14	знайти значення максимального кратного 3 числа масиву
15	Записати число 100 замість максимального позитивного елемента
16	Знайти значення максимального кратного 3 числа масиву і його номер
17	Знайти значення мінімального не кратна 3 числа масиву і його номер
18	У масиві поміняти місцями найбільший та найменший елементи
19	Знайти значення мінімального кратного 3 числа масиву

Продовження таблиці 3.3

1	2
20	Знайти частку від ділення мінімального на максимальний елементів масиву
21	Знайти номер мінімального числа масиву, що належить інтервалу [-3, 4]
22	Знайти номер максимального числа масиву, що належить інтервалу [8, 25]
23	Знайти значення мінімального числа масиву, що належить інтервалу [3, 12]
24	Знайти значення максимального числа масиву, що належить інтервалу [-5, 15]
25	Знайти номер і значення найбільшого негативного парного елемента масиву

Завдання 2. Скласти блок-схему й програму для оброблення масиву медичних даних.

2.1 Нехай досліджуваною сукупністю є новонароджені у великому місті за одну добу хлопчики, а ознакою *x* є вага (таблиця 3.4).

Таблиця 3.4 – Вага новонароджених

x_i	2,7	2,8	2,9	3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9	4,0	4,1	4,2	4,3	4,4

Знайти середнє, максимальне та мінімальне значення виміряних даних ваги.

2.2 Нехай досліджуваною сукупністю є дані про вільні місця в лікувальних закладах міста. (таблиця 3.5).

Таблиця 3.5 – Дані про вільні місця у ЛЗ

n_i	1	2	3	7	8	12	13	10	7	6	5	6	6	5	3	3	2	1

Знайти середнє, максимальне та мінімальне значення статистичних даних.

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв'язання задачі; програму;

3) рішення задачі (лістинг) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. На кого або на що покладається контроль меж числових масивів у мові програмування **C++**?

2. Для чого застосовується початкова ініціалізація числових масивів при подальшому їх використанні?

3. Як проводиться пошук максимального та мінімального елементів масиву?

4. Що визначає індекс елемента в масиві?

3.5 Формування робочих масивів за допомогою операцій селекції вихідного масиву

До робочого масиву з вихідного вибираються тільки ті елементи, які задовольняють поставленій умові. Розглянемо на прикладі.

Приклад 14. Сформувати новий масив із позитивних елементів вихідного масиву *x(15)*.

Порядок роботи:

Крок 1. Вводимо масив *x(15)*.

Крок 2. Установлюємо вихідний індекс робочого масиву $j = 0$.

Крок 3. Організовуємо цикл, що перебирає елементи вихідного масиву (тобто індекс *i*), починаючи з 0-го і кінчаючи 14-м.

Крок 4. Якщо $x_i \le 0$, то йдемо на крок 7.

Крок 5. Встановлюємо індекс наступного елемента робочого масиву $j = j + 1$.

Крок 6. Привласнюємо елементу робочого масиву значення елемента вихідного масиву $y_j = x_i$.

Крок 7. Якщо цикл по *i* не закінчився, йдемо на початок циклу, тобто на крок 3.

Крок 8. Друкуємо *j* елементів робочого масиву *y*.

Крок 9. Кінець алгоритму.

3.6 Лабораторна робота № 8. Формування одновимірного робочого масиву

Мета роботи: вивчити роботу з масивом як зі складеним типом даних, прийоми введення й виведення даних, оброблення одновимірних масивів; вивчити і навчитися застосовувати оброблення масивів за заданими логічними умовами, формувати нові масиви.

Завдання. Скласти блок-схему і програму для оброблення масиву відповідно до індивідуального завданням, наведеного в таблиці 3.6. Розмір і ім'я масиву вибрати самостійно.

Таблиця 3.6 – Індивідуальні завдання

Вар.	**Умова задачі**
1	2
1	Даний масив *X(15)*. Сформувати новий масив із парних елементів вихідного
2	Даний масив *X(25)*. Сформувати новий масив із непарних елементів вихідного
3	Даний масив *D(15)*. Сформувати новий масив із кратних 3 елементів вихідного

Продовження таблиці 3.6

1	2
4	Даний масив *A(10)*. Сформувати новий масив із негативних елементів вихідного
5	Даний масив *Z(15)*. Сформувати новий масив із позитивних парних елементів вихідного
6	Даний масив *X(25)*. Сформувати новий масив з елементів вихідного, лежачих в інтервалі [-3,7]
7	Даний масив Y(10). Сформувати новий масив із непарних позитивних елементів вихідного
8	Даний масив D(12). Сформувати новий масив із позитивних кратних 3 елементів вихідного
9	Даний масив A(8). Сформувати новий масив із негативних парних елементів вихідного
10	Даний масив C(15). Сформувати новий масив з елементів вихідного, які більше 8
11	Даний масив B(21). Сформувати новий масив із кратних 4 елементів вихідного
12	Даний масив A(12). Сформувати новий масив із негативних непарних елементів вихідного
13	Даний масив X(8). Сформувати новий масив із негативних не кратних 3 елементів вихідного
14	Даний масив G(9). Сформувати новий масив із парних елементів вихідного масиву, що стоять на непарних місцях
15	Даний масив Y(15). Сформувати новий масив із непарних, кратних 3 елементів вихідного
16	Даний масив A(18). Сформувати новий масив із непарних, кратних 5 елементів вихідного
17	Даний масив Z(10). Сформувати новий масив із парних елементів вихідного, лежачих в інтервалі [1,12]
18	Даний масив A(11). Сформувати новий масив із непарних елементів вихідного, лежачих в інтервалі [-3,15]
19	Даний масив B(10). Сформувати новий масив із номерів негативних парних елементів вихідного
20	Даний масив A(8). Сформувати новий масив з номерів негативних непарних елементів вихідного
21	Даний масив C(12). Сформувати новий масив із негативних елементів вихідного, що стоять на парних місцях
22	Даний масив F(13). Сформувати новий масив із негативних елементів вихідного, що стоять на непарних місцях
23	Даний масив H(12). Сформувати новий масив із позитивних елементів вихідного, що стоять на парних місцях

Продовження таблиці 3.6

1	2
24	Даний масив V(19). Сформувати новий масив із негативних елементів вихідного, лежачих у діапазоні [-20, -5]
25	Даний масив N(11). Сформувати новий масив із негативних кратних 5 елементів вихідного
26	Даний масив K(15). Сформувати новий масив із позитивних елементів вихідного, що стоять на непарних місцях
27	Даний масив Y(11). Сформувати новий масив із негативних не кратних 5 елементів вихідного
28	Даний масив Z(14). Сформувати новий масив із позитивних кратних 5 елементів вихідного
29	Даний масив R(13). Сформувати новий масив із негативних кратних 10 елементів вихідного
30	Даний масив N(11). Сформувати новий масив із негативних кратних 8 елементів вихідного

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв'язання задачі; програму;

3) рішення задачі (лістинг) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. Які класи пам'яті можна використовувати при оголошенні масиву?

2. Які класи пам'яті використовуються за умовчанням?

3. Як розміщаються елементи масиву в пам'яті?

4. Чи існують операції роботи з масивами, як об'єктами?

5. У яких випадках розмірність масиву при оголошенні можна не вказувати?

3.7 Сортування елементів масиву

Сортування – це впорядкування множини спаданням або зростанням ознаки, загального для всіх елементів множини. Існує кілька методів сортування. Якщо треба впорядкувати частину масиву, що задовольняє заданим умовам, то спочатку відбирають потрібні елементи в робочий масив, а потім його сортують. Якщо робочий масив не сформувався, то виводять про це повідомлення і зупиняють програму. Розглянемо на прикладі.

Приклад 15. Знайти суму двох найбільших негативних елементів масиву *x(15)*.

Порядок роботи:

Крок 1. Вводимо масив *x(15)*.

Крок 2. Установлюємо вихідний індекс робочого масиву $j = 0$.

Крок 3. Організовуємо цикл, що перебирає елементи вихідного масиву (тобто індекс *i*), починаючи з 0-го і закінчуючи 14-м.

Крок 4. Якщо $x_i \geq 0$, то йдемо на крок 7.

Крок 5. Встановлюємо індекс наступного елемента робочого масиву $j = j + 1$.

Крок 6. Привласнюємо елементу робочого масиву значення елемента вихідного масиву $y_j = x_i$.

Крок 7. Якщо цикл за *i* не закінчився, йдемо на початок циклу, тобто на крок 3.

Крок 8. Якщо $j < 2$, то видаємо повідомлення «Масив не сформований» і йдемо на крок 17.

Крок 9. Організовуємо цикл, що визначає кількість переглядів робочого масиву (тобто індекс *k*), починаючи з 0-го і закінчуючи *(j-1)*-м.

Крок 10. Організовуємо цикл, що визначає пару елементів робочого масиву, що переглядається, (тобто індекс *i*), починаючи з 0-го і закінчуючи *(j-2)*-м.

Крок 11. Якщо перший елемент не більше другого, то йдемо на крок 13.

Крок 12. Змінюємо два елементи місцями:

$$c = y_i;\ y_i = y_{i+1};\ y_{i+1} = c.$$

Крок 13. Якщо цикл по *i* не закінчився, йдемо на початок циклу, тобто на крок 10.

Крок 14. Якщо цикл за *k* не закінчився, йдемо на початок циклу, тобто на крок 9.

Крок 15. Обчислюємо суму $s = y_j + y_{j-1}$.

Крок 16. Друкуємо *s*.

Крок 17. Кінець алгоритму.

Для демонстрації деяких особливостей вкладення циклів і роботи з масивами розглянемо найпростіші алгоритми сортування. Необхідно, ввівши значення змінної 1 < n <= 100 і значення n перших елементів масиву a [0], a [1], ..., a [n-1], впорядкувати ці перші елементи масиву за зростанням їхніх значень. Текст першого варіанта програми:

```
/ * Упорядкування елементів масиву * /
 # include <stdio.h>
main ()
 {
int n, i, j;
double a [100], b;
while (1)
 {
printf ( "\ n Введіть кількість елементів n =");
scanf ( "% d", & n);
if (n> 1 && n <= 100) break;
printf ( "Помилка! Необхідно 1 <n <= 100!");
```

```
}
printf ( "\ n Введіть значення елементів масиву: \ n");
for (j = 0; j <n; j ++)
 {
printf ( "a [% d] =", j + 1);
scanf ( "% lf ", □a [j]);
}
for (i = 0; i <n-1; i ++)
for (j = i + l; j < n; j ++)
if (a [i]> a [j])
 {
b = a [i]; a [i] = a [j] ;. a [j] = b;
}
printf ( "\ n Впорядкований масив: \ n");
for (j = 0; j <n; j ++)
printf ( "a [% d] =% f \ n", j + 1, a [j]);
}
```

При заповненні масиву і при друку результатів його впорядкування індексація елементів виконана від 1 до *n*, як це зазвичай прийнято в математиці. У програмі на Сі це відповідає зміні індексу від 0 до *(n-1)*.

У програмі реалізований алгоритм прямого упорядкування – кожен елемент *a [i]*, починаючи з *a [0]* і закінчуючи *a [n-2]*, порівнюється з усіма подальшими, і на місце *a [i]* вибирається мінімальний. Таким чином, *a [0]* приймає мінімальне значення, *a [1]* – мінімальне з решти і т. д. Недолік цього алгоритму полягає в тому, що в ньому фіксоване число порівнянь, незалежне від вихідного розташування значень елементів. Навіть для вже упорядкованого масиву доведеться виконати ту ж саму кількість ітерацій *(n-1) * n / 2*, оскільки умови закінчення циклів не пов'язані з властивостями, тобто з розміщенням елементів масиву.

Алгоритм попарного порівняння сусідніх елементів дозволяє в ряді випадків зменшити кількість ітерацій при упорядкуванні. У циклі від 0 до *n-2* кожен елемент *a [i]* масиву порівнюється з наступним *a [i + l] (0 <i <n-l).* Якщо *a [i]> a [i + l],* то значення цих елементів міняються місцями. Впорядкування закінчується, якщо виявилося, що *a [i]* не більш *a [i + l]* для всіх *i.* Нехай *k* – кількість перестановок при черговому перегляді. Тоді впорядкування можна здійснити за допомогою такої послідовності операторів:

```
.....
do {
for (i=0, k=0; i<n-l; i++)
   if ( a[i] > a[i+l]  )
    {
      b=a[i]; a[i]=a[i+1]; a[i+1]=b;
     k=k+l;
}
n--;
}
while ( k > 0 ) ;
....
```

Тут кількість повторень зовнішнього циклу залежить від вихідного розташування значень елементів масиву. Після першого завершення внутрішнього циклу елемент *a [n-1]* стає максимальним. Після другого закінчення внутрішнього циклу на місце *a [n-2]* вибирається максимальний з решти елементів і т. д. Таким чином, після *j*-го виконання внутрішнього циклу елементи *a [n-j], ..., a [n-l]* вже впорядковані, і наступний внутрішній цикл досить виконати тільки для *0 < i <(n-j-l).* Саме тому після кожного закінчення внутрішнього циклу значення n зменшується на 1.

У разі впорядкованості вихідного масиву зовнішній цикл повторюється тільки один раз, при цьому виконується *(n-1)* порівнянь, *k* залишається рівним 0. Для випадку, коли вихідний масив впорядкований за спадан-

ням, кількість ітерацій зовнішнього циклу одно *(n-1)*, а внутрішній цикл послідовно виконується *(n-1) * n / 2* раз.

3.7.1 Метод обмінного сортування

Метод обмінного сортування (бульбашковий) ще називається «бульбашковим» методом, один з найпростіших методів сортування. Основні позитивні риси методу – це легка реалізація у вигляді програми. Алгоритм складається в повторюваних проходах посортованого масиву. За кожен прохід елементи послідовно порівнюються попарно і, якщо порядок в парі невірний, виконується обмін елементів. Проходи по масиву повторюються до тих пір, поки на черговому проході не опиниться, що обміни більше не потрібні, що означає – масив відсортований. При проході алгоритму, елемент, що стоїть не на своєму місці, «спливає» до потрібної позиції як бульбашка у воді, звідси і назва алгоритму.

Алгоритм сортування методом бульбашки:

1. Порівнюємо перший і другий елементи масиву. Якщо перший елемент більший, ніж другий, то міняємо їх місцями.

2. Порівнюємо другий і третій елементи масиву. Якщо другий елемент більший, ніж третій, то міняємо їх місцями.

3. ... 4. Порівнюємо передостанній (N-1) і останній (N) елементи масиву. Якщо передостанній елемент більший, ніж останній, то міняємо їх місцями.

Програма, що реалізує представлений алгоритм, наведена на рис. 3.2.

```
#include <iostream>
#define M 10
void main()
{
        float mas[M], temp;
        for(Int i = 0; i <M; ++ i)
                cin >> mas[i];
        bool loop = true;
        int count = M – 1;
        while (loop)
        {
                loop = false;
                for (int i = 0; i <count; ++ i)
                        if (mas0[i]> mas [i + 1])
                        {
                                temp = mas[i];
                                mas[i] = mas [i + 1];
                                mas[i + 1] = temp;
                                loop = true;
                        }
                count–;
        }
        for (int i = 0; i <M; ++ i)
                cout << mas[i] << '\ n';
}
```

Рисунок 3.2 Реалізація «бульбашкового» методу

У програмі сортування проводиться в масиві *mas [M]*. У результаті роботи програми елементи масиву будуть впорядковані за зростанням їхніх значень.

Слід також зазначити, що в результаті кожного перебору масиву максимальний елемент стає на своє місце. Тому для скорочення числа операцій використовується змінна **count**, яка має початкове значення, рівне розміру масиву, але після виконання кожного проходу зменшується на 1, зменшуючи тим самим довжину масиву.

У найгіршому разі упорядковуватися буде масив, що складається з двох елементів. Може 10 скластися ситуація, коли масив довжиною більше двох елементів вже впорядкований. При проході за таким масиву не буде виконано жодного обміну, і процедуру сортування можна зупинити. Для визначення такої ситуації використовується логічна змінна **loop.**

3.7.2 Метод сортування злиттям

Метод сортування злиттям. При сортуванні злиттям на *i*-му кроці масив a розбивається на впорядковані підмасиви (послідовності елементів масиву, що розташовані поспіль) довжини *Size*. Пари сусідніх підмасивів зливаються у впорядковані підмасиви довжини *Size*2* у допоміжному масиві *b*. Після присвоєння a значення b та збільшення вдвічі значення *Size* злиття повторюється для підмасивів більшого розміру. Процес завершується, коли *Size* стає більшим або рівним *n*. Оскільки при *i = 1* підмасиви з 1 елемента впорядковані за означенням, у результаті буде отримано впорядкований масив a.

3.8 Лабораторна робота № 9. Сортування одновимірних масивів. Сортування одновимірних масивів медичних даних

Мета роботи: вивчити і навчитися застосовувати оброблення масивів за заданими логічними умовами, формувати нові масиви й сортувати їх.

Завдання 1. Скласти блок-схему і програму для оброблення масиву відповідно до індивідуального завдання, наведеного в таблиці 3.7. Розмір й ім'я масиву вибрати самостійно.

Таблиця 3.7 – Індивідуальні завдання

Вар.	**Умова задачі**
1	Знайти суму двох найбільших парних елементів масиву
2	Знайти добуток двох найбільших непарних елементів масиву
3	Знайти добуток двох найбільших парних елементів масиву
4	Знайти суму двох найбільших непарних елементів масиву
5	Знайти суму трьох найбільших парних елементів масиву
6	Знайти суму двох найменших парних елементів масиву
7	Знайти суму двох найменших непарних елементів масиву
8	Знайти суму трьох найменших непарних елементів масиву
9	Знайти суму двох найменших позитивних елементів масиву
10	Знайти суму двох найбільших негативних елементів масиву
11	Знайти суму трьох найменших позитивних елементів масиву
12	Знайти добуток двох найменших позитивних елементів масиву
13	Знайти добуток двох найбільших негативних елементів масиву
14	Знайти добуток трьох найбільших кратних 5 елементів масиву
15	Знайти добуток трьох найменших не кратних 4 елементів масиву
16	Знайти добуток трьох найбільших позитивних кратних 3 елементів масиву
17	Знайти добуток трьох найменших негативних непарних елементів масиву
18	Знайти суму трьох найменших позитивних парних елементів масиву
19	Знайти суму трьох найбільших непарних, лежачих в інтервалі [1, 30], елементів масиву
20	Знайти добуток чотирьох найменших, лежачих в інтервалі [-20, 20], елементів масиву
21	Знайти суму чотирьох найменших кратних 5 і не більше 50 елементів масиву
22	Знайти добуток двох найбільших і двох найменших позитивних парних елементів масиву
23	Знайти суму двох найбільших і двох найменших негативних парних елементів масиву
24	Знайти добуток двох найбільших і двох найменших негативних непарних елементів масиву
25	Знайти суму двох найбільших і двох найменших непарних елементів масиву, що лежать в інтервалі [1, 25]

Завдання 2. Сортування одновимірних масивів медичних даних для діагностики деяких гострих станів при наявності симптомів хвороби. Масив даних ввести з клавіатури.

Варіант № 1. Масив даних складається з 10 дійсних елементів, значення яких характерні для: інфаркту міокарда: частота серцевого ритму ≥ 100/хв. Впорядкувати елементи масиву по спадаючій та указати під номером першого елемента – прізвище хворого, який потребує негайної допомоги. Масив даних ввести з клавіатури.

Варіант № 2. Масив даних складається з 10 дійсних елементів, значення яких характерні для інфаркту міокарда: систолічний артеріальний тиск ≥ 150 мм рт.ст.. Впорядкувати елементи масиву по спадаючій та указати під номером першого елемента – прізвище хворого, який потребує негайної допомоги.

Варіант № 3. Масив даних складається з 10 дійсних елементів, значення яких характерні для інфаркту міокарда: фібриляція шлуночків > 300/хв. Впорядкувати елементи масиву по спадаючій та указати під номером першого елемента – прізвище хворого, який потребує негайної допомоги.

Варіант № 4. Масив даних складається з 10 дійсних елементів, значення яких характерні для інфаркту міокарда: надшлуночкова тахікардія (ритм з частотою > 100/хв.) Впорядкувати елементи масиву по спадаючій та указати під номером першого елемента – прізвище хворого, який потребує негайної допомоги.

Варіант № 5. В одновимірному масиві даних про захворюваність дітей у віці до 3 років: підвищення температури тіла > 38°С, що складається з 15 дійсних елементів: Перетворити масив таким чином, щоб у першій його половині розташовувалися елементи, що стояли в інтервалі [39–40.3], а потім – всі інші.

Варіант № 6. В одновимірному масиві даних про захворюваність дітей у віці до 3 років: підвищення частоти серцевого ритму ≥ 170/хв., що

складається з 15 дійсних елементів: Перетворити масив таким чином, щоб в першій його половині розташовувалися елементи, що стояли в інтервалі [180–190], а потім – усі інші.

Варіант № 7. В одновимірному масиві даних про захворюваність дітей у віці до 3 років: брадикардія, частота серцевого ритму ≤ 90/хв., що складається з 15 дійсних елементів: Перетворити масив таким чином, щоб в першій його половині розташовувалися елементи, що стояли в інтервалі [70–80], а потім – усі інші.

Варіант № 8. В одновимірному масиві, що констатує підвищення цукру в крові, складається з 20 дійсних елементів: Стиснути масив, видаливши з нього всі елементи, значення яких перевищує показник 8,0. Вивільнені в кінці масиву елементи заповнити нулями.

Варіант № 9. В одновимірному масиві, що констатує різке зниження рівня цукру в крові, складається з 20 дійсних елементів: Стиснути масив, видаливши з нього всі елементи, значення яких не перевищує 3. Вивільнені в кінці масиву елементи заповнити нулями.

Варіант № 10. В одновимірному масиві про підвищення рівня тромбоцитів в крові від 650 000 до 800 000 , складається з 15 дійсних елементів: Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, значення яких лежить в інтервалі [700 000, 750 000], а потім – усі інші.

Варіант № 11. В одновимірному масиві, що показує зниження рівню гемоглобіну в крові пацієнтів із травмами, який складається з 15 дійсних елементів, перетворити масив таким чином, щоб елементи, рівні 120 g / L, розташовувалися після всіх інших.

Варіант № 12. Масив даних складається з 10 дійсних елементів, значення яких характерні для інфаркту міокарда: систолічний артеріальний тиск ≥ 150 мм рт. ст. Впорядкувати елементи масиву по зростанню та указати під номером останнього елемента – прізвище хворого, який потребує негайної допомоги.

Варіант №1 3. У масиві зберігаються дані про підвищення температури тіла за тиждень хворих на вірусну інфекцію. Знайти кількість днів, де відхилення температури не більше 10 % від норми.

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв'язання задачі; програму;

3) рішення задачі (лістинг) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. Що таке сортування масиву?

2. Які є види сортування?

3. Які операції допустимі над елементами масиву та над масивом в цілому?

4. Селективне оброблення елементів масиву (парність, непарність тощо).

5. Відмінність сортування методом бульбашки і методом вибору, який із них швидше.

4 ПОНЯТТЯ БАГАТОВИМІРНОГО МАСИВУ МЕДИЧНИХ ДАНИХ. ВКЛАДЕНІ ЦИКЛИ. УПОРЯДКУВАННЯ В ОДНОМІРНИХ МАСИВАХ. ПЕРЕМИКАЧІ. АЛЬТЕРНАТИВНИЙ ВИБІР. ОБРОБЛЕННЯ ЕЛЕМЕНТІВ МАТРИЦЬ

4.1 Багатовимірні масиви

Правила ініціалізації багатовимірних масивів відповідають визначенню багатовимірного масиву як одновимірного, елементами якого служать масиви, розмірність яких на одиницю менше, ніж у вихідного масиву. Одновимірна масив ініціалізується укладеним у фігурні дужки списком початкових значень. У свою чергу, початкове значення, якщо воно відноситься до масиву, також є укладений у фігурні дужки список початкових значень. Наприклад, привласнити початкові значення речовим елементів двовимірного масиву A, що складається з трьох "рядків" і двох "стовпців", можна наступним чином:

double A[3][2]={{10,20}, (30,40), {50,60}};

Цей запис еквівалентний послідовності операторів присвоювання: A [0] [0] = 10; A [0] [1] = 20; A [1] [0] = 30; A [1] [1] = 40; A [2] [0] = 50; A [2] [1] = 60 ;. Той же результат можна отримати з одним списком ініціалізації:

double A[3][2]={10,20,30,40,50,60};

За допомогою ініціалізацій можна присвоювати значення не всім елементам багатовимірного масиву. Наприклад, щоб ініціалізувати тільки елементи першого стовпця матриці, її можна описати так:

double Z [4] [6] = {{1}, {2}, (3), {4} };

Наступний опис формує "трикутну" матрицю в цілочисловому масиві з 5 рядків і 4 стовпців:

int x [5] [4] = {{ 1), {2,3}, {4,5,6}, {7,8,9,10}};

У цьому прикладі останній п'ятий рядок *x* [4] залишається незаповненим. Перші три рядки заповнені не до кінця.

4.2 Вкладені цикли

У тілі циклу дозволені будь-які виконані оператори, в тому числі і цикли, тобто можна конструювати вкладені цикли. Вкладені цикли використовуються при роботі з багатомірними функціями і множинами. При цьому потрібно стежити, щоб не було перехрещування циклів.

Приклад 16. Як приклад розглянемо фрагмент програми для отримання суми такого виду: $s = \sum_{j=1}^{10}\left[\prod_{i=1}^{5} a_{ji} + \sum_{i=1}^{5} a_{ji}\right]$

Введемо такі позначення: *a* – двовимірний масив, що містить значення елементів матриці; *p* – добуток елементів рядка матриці; *c* – сума їхніх значень; *s* – шукана сума (результат). Опустивши визначення змінних й оператори введення-виведення, запишемо текст на мові Сі:

```
double a[10][5];
for( s=0.0,j=0; j<10; j++)
{
   for( p=1.0,c=0.0,i=0; i<5; i++)
       {
         p*=a[j][i];
         c+=a[j][i];
        }
s+=c+p;
}
```

При роботі з вкладеними циклами слід звернути увагу на правила виконання операторів **break** і **continue**. Кожен з них діє тільки в тому опе-

раторі, в тілі якого він безпосередньо використаний. Оператор **break** припиняє виконання найближчого зовнішнього для нього оператора циклу. Оператор **continue** передає управління на найближчу зовнішню перевірку умови продовження циклу.

Для ілюстрації розглянемо фрагмент іншої програми для обчислення суми добутків елементів рядків тієї ж матриці:

$$s = \sum_{j=1}^{10} \prod_{i=1}^{5} a_{ji}$$

```
double a[10][5];
for (j=0,s=0.0; j<10; j++)
 {
    for (i=0,p=1.0; i<5; i++)
     {
       if (a[j][i] ==0.0) break;
       p*=a[j][i];
     }
    if (i <5)  continue;
     s+=p;
  }
```

Внутрішній цикл переривається, якщо виявляється нульовий елемент матриці. У цьому випадку твір елементів стовпця свідомо дорівнює нулю, і його не потрібно обчислювати. У зовнішньому циклі перевіряється значення. Якщо $i < 5$, тобто елемент *a [j] [i]* виявився нульовим, то оператор **continue** передає управління на найближчий оператор циклу, і, таким чином, не відбувається збільшення s на величину "недорахувалися" значення *p*. Якщо внутрішній цикл завершено природно, то дорівнює 5, і оператор **continue** не може виконуватися.

Розглянемо приклади запису алгоритмів оброблення двовимірних масивів на прикладах.

Приклад 17. У матриці *a(4,5)* знайти добуток позитивних елементів. Блок-схема алгоритму приведена на рисунку 4.1.

Порядок роботи:

Крок 1. Уводимо матрицю *a(4,5)*.

Крок 2. Задаємо початкове значення добутку $p = 1$.

Крок 3. Організовуємо цикл, що перебирає рядки матриці (тобто індекс *i*), починаючи з 0-го і закінчуючи 3-м.

Крок 4. Організуємо цикл, що перебирає стовпці матриці (тобто індекс *j*), починаючи з 0-го і закінчуючи 4-м.

Крок 5. Якщо $a_{ij} > 0$, тоді привласнюємо $p = p \cdot a_{ij}$.

Крок 6. Якщо цикл за *j* не закінчився, йдемо на початок циклу, тобто на крок 4.

Крок 7. Якщо цикл за *i* не закінчився, йдемо на початок циклу, тобто на крок 3.

Крок 8. Друкуємо *p*.

Крок 9. Кінець алгоритму

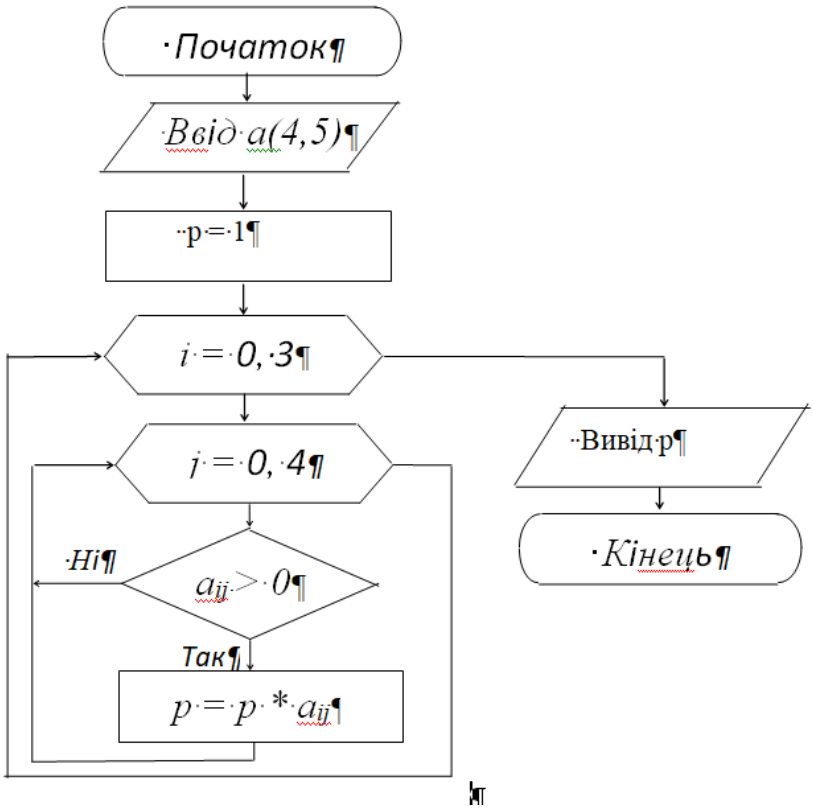


Рисунок 4.1 Блок-схема алгоритму

Приклад 18. Знайти суму парних елементів третього рядка матриці *a(4,5)*.

Порядок роботи:

Крок 1. Уводимо матрицю *a(4,5)*.

Крок 2. Установлюємо початкове значення суми *s = 0*.

Крок 3. Задаємо номер оброблюваного рядка *i = 2*.

Крок 4. Організовуємо цикл, що перебирає стовпці матриці (тобто індекс *j*), починаючи з 0-го і кінчаючи 4-м.

Крок 5. Якщо a_{ij} парне, тоді привласнюємо $s = s + a_{ij}$.

Крок 6. Якщо цикл за *j* не закінчився, йдемо на початок циклу, тобто на крок 4.

Крок 7. Друкуємо *s*.

Крок 8. Кінець алгоритму.

Приклад 19. Знайти суму парних елементів головної діагоналі матриці *a(5,5)*.

Порядок роботи:

Крок 1. Уводимо матрицю *a(5,5)*.

Крок 2. Установлюємо початкове значення суми *s = 0*.

Крок 3. Організовуємо цикл, що перебирає рядки матриці (тобто індекс *i*), починаючи з 0-го і закінчуючи 4-м.

Крок 4. Організовуємо цикл, що перебирає стовпці матриці (тобто індекс *j*), починаючи з 0-го і закінчуючи 4-м.

Крок 5. Якщо a_{ij} парне й $i = j$, привласнюємо $s = s + a_{ij}$.

Крок 6. Якщо цикл за *j* не закінчився, йдемо на початок циклу , тобто на крок 4.

Крок 7. Якщо цикл за *i* не закінчився, йдемо на початок циклу, тобто на крок 3.

Крок 8. Друкуємо *s*.

Крок 9. Кінець алгоритму.

4.3 Лабораторна робота № 10. Селективне оброблення двовимірних масивів

Мета роботи: вивчити конструкції мови С й оператори для оброблення багатовимірних масивів із застосуванням оператора циклу **for;** виробити практичні навички в складанні алгоритмів на селективне оброблення двовимірних масивів.

Завдання. Скласти блок-схему і програму для оброблення багатовимірних масивів. Індивідуальні завдання наведені в таблиці 4.1. Ім'я та розмір матриці вибрати самостійно.

Таблиця 4.1 – Індивідуальні завдання

Вар.	**Умова задачі**
1	Знайти добуток негативних елементів матриці
2	Знайти мінімальний за модулем елемент матриці
3	Знайти максимальний парний елемент матриці
4	Знайти мінімальний непарний елемент матриці
5	Обчислити суму позитивних і суму негативних елементів матриці
6	Знайти середнє арифметичне позитивних елементів матриці
7	Знайти середнє геометричне позитивних елементів матриці
8	Знайти кількість елементів матриці, що знаходяться в інтервалі (-3, 8]
9	Знайти середнє арифметичне елементів матриці, які більше 5
10	Знайти добуток парних елементів матриці
11	Знайти середнє арифметичне негативних елементів матриці
12	Знайти суму добутків позитивних елементів рядків матриці
13	Знайти суму добутків негативних елементів стовпців матриці
14	Знайти добуток сум негативних елементів рядків матриці

Продовження таблиці 4.1

Вар.	Умова задачі
15	Знайти добуток сум позитивних елементів стовпців матриці
16	Знайти суму добутків елементів рядків матриці
17	Знайти суму непарних елементів стовпців матриці
18	Знайти добуток непарних елементів рядків матриці
19	Знайти добуток сум парних елементів стовпців матриці
20	Знайти максимальний негативний елемент матриці
21	Знайти мінімальний позитивний елемент матриці
22	Визначити місце мінімального парного елемента матриці
23	Знайти усі негативні елементи матриці
24	Знайти усі позитивні елементи матриці
25	Знайти усі негативні парні елементи матриці
26	Визначити місце максимального позитивного елемента матриці
27	Знайти усі позитивні непарні елементи матриці
28	Знайти середнє арифметичне парних елементів матриці
29	Знайти середнє геометричне непарних елементів матриці
30	Визначити рядок, у якому знаходиться мінімальний за модулем елемент матриці

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв'язання задачі; програму;

3) рішення задачі (лістинг) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. Як організовуються багатовимірні числові масиви в мові Сі?

2. Як організується індексування числових масивів у мові Сі?

3. В якій послідовності й як відбувається заповнення багатовимірних числових масивів у програмах на мові Сі?

4.4 Ініціалізація масивів

При ініціалізації багатовимірного масиву для покращення наочності елементи ініціалізації кожного вимірювання можна укладати у фігурні дужки.

Приклад ініціалізації двовимірного масиву:

```
int  MN[3][4] = {
{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12}
};
```

Масив MN[3][4] – це матриця, у якій 3 рядки й 4 стовпці.

Для багатовимірних масивів ініціалізацію можна також проводити із зазначенням номера ініціалізованого елемента.

Приклад ініціалізації тривимірного масиву:

```
int  XYZ[2][3][4] = {
{ {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} },
{ {13, 14, 15, 16}, {17, 18, 19, 20}, {21, 22, 23, 24} }
};
```

Як видно, масив XYZ містить два блоки, кожен з яких є матриця розміру 3 × 4, тобто 3 рядки й 4 стовпці.

У багатовимірному масиві розмір самого лівого вимірювання також можна не вказувати. Зокрема, для ініціалізації масиву MN[3][4] допустимий такий запис:

```
int  MN[][4] = {
{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12}
};
```

При ініціалізації багатовимірних масивів необхідно вказати всі дані (розмірності) за винятком крайньої зліва розмірності. Це потрібно для того, щоб компілятор зміг визначити довжину підмасивів, які складають масив, і зміг виділити необхідну пам'ять. Розглянемо приклад безрозмірної ініціалізації для тривимірного масиву цілих чисел:

```
int  XYZ[][3][4] = {
{
{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12}
},
{
{13, 14, 15, 16},
{17, 18, 19, 20},
{21, 22, 23, 24}
}
};
```

Виведення тривимірного масиву на консоль (дисплей) можна виконати за такою програмою:

```
#include <stdio.h>
#include <conio.h>
#include <locale.h>
int main (void) {
int i, j, k;
```

```
setlocale(LC_ALL, «Russian»);
int  XYZ[][3][4] = {
{{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} }, // 1-й
{{13, 14, 15, 16}, {17, 18, 19, 20}, {21, 22, 23, 24} } }; // 2-й
        for (i = 0; i < 2; ++i) { printf("\n»);
                for (j = 0; j < 3; ++j) { printf("\n»);
                        for (k = 0; k < 4; ++k)
                                printf(» %3d», XYZ[i][j][k]);
                }
        }
printf("\n\n Натисніть будь-яку клавішу:");
        _getch();
        return 0;}
```

4.5 Оброблення елементів матриць

При обробленні двовимірних масивів часто доводиться виділяти елементи:

- *k – й рядка* *A[i][j], де i==k, j = 0,…, M-1*
- *k – го стовпця* *A[i][j], де i=0, …, N-1; j==k*

а для квадратних матриць (M = N) також:

- *головної діагоналі* *A[i][i], де i = 0, …, N-1*
- *побічної діагоналі* *A[i][N-1-i], де i = 0, …, N-1*
- *над головною діагоналлю* *A[i][j], де i < j*
- *під головною діагоналлю* *A[i][j], де i > j*

4.6 Лабораторна робота № 11. Оброблення рядків і стовпчиків матриць

Мета роботи: вивчити конструкції мови С й оператори для оброблення багатовимірних масивів із застосуванням оператора циклу **for;** виробити практичні навички в складанні алгоритмів на оброблення заданої рядки / стовпці матриці.

Завдання. Скласти блок-схему й програму для оброблення багатовимірних масивів. Індивідуальні завдання наведені в таблиці 4.2. Ім'я та розмір матриці вибрати самостійно.

Таблиця 4.2 – Індивідуальні завдання

Вар.	Умова задачі
1	Знайти суму парних елементів четвертого рядка матриці
2	Знайти суму непарних елементів третього стовпця матриці
3	Обчислити добуток негативних елементів другого стовпця матриці
4	Визначити кількість негативних елементів другого рядка матриці
5	Знайти добуток непарних елементів четвертого стовпця матриці
6	Знайти добуток позитивних елементів другого рядка матриці
7	Знайти суму непарних елементів п'ятого рядка матриці
8	Знайти суму позитивних парних елементів другого стовпця матриці
9	Знайти максимальне негативне число третього стовпця матриці
10	Знайти мінімальне позитивне число четвертого рядка матриці
11	Знайти максимальне парне число п'ятого рядка матриці
12	Знайти мінімальне непарне число четвертого стовпця матриці
13	Знайти суму позитивних кратних 5 елементів другого стовпця матриці
14	Знайти кількість негативних не кратних 3 елементів п'ятого рядка матриці
15	Знайти кількість позитивних парних елементів третього рядка матриці
16	Знайти добуток квадратів тих елементів третього стовпця матриці, модулі яких належать інтервалу [1, 30]
17	Знайти середнє арифметичне негативних елементів сьомого рядка матриці
18	Знайти суму ненульових елементів другого стовпця матриці

Продовження таблиці 4.2

Вар.	Умова задачі
19	Знайти різницю сум негативних і позитивних елементів другого стовпця матриці
20	Знайти середнє геометричне модулів негативних елементів п’ятого стовпця матриці
21	Знайти кратні семи позитивні елементи третього рядка матриці
22	Знайти середнє арифметичне позитивних елементів другого рядка матриці
23	Знайти середнє геометричне кратних 3 елементів п’ятого рядка матриці
24	Знайти частку від розподілу кількості негативних елементів п’ятого рядка матриці на їхню суму
25	Знайти добуток ненульових елементів п’ятого стовпця матриці
26	Знайти середнє арифметичне кратних 5 елементів п’ятого рядка матриці
27	Знайти середнє геометричне ненульових елементів п'ятого стовпця матриці
28	Знайти добуток кратних 7 елементів першого стовпця матриці
29	Знайти частку від розподілу суми негативних елементів другого рядка матриці на їхній добуток
30	Знайти квадрат мінімального елемента третього рядка матриці

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв'язання задачі; програму;

3) рішення задачі (лістинг) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. Приведіть приклади оголошення масивів з різною розмірністю. Поясніть організацію розміщення елементів масиву у пам’яті?

2. Які синтаксичні конструкції можна використовувати для доступу до елементів масиву?

3. Приведіть загальну формулу для масиву **<Тип> <Ім'я> [N][M][K]** при доступі до заданого елемента **<Ім'я> [i][j][k]** і поясніть її?

4. Як здійснюється ініціалізація багатомірних масивів?

5. Як відбувається відбір елементів за рядками, стовпцями, діагоналями?

6. Селективний відбір елементів матриці.

4.7 Лабораторна робота № 12. Оброблення квадратної матриці

Мета роботи: вивчити конструкції мови С і оператори для оброблення багатовимірних масивів із застосуванням оператора циклу **for**; виробити практичні навички в складанні алгоритмів на оброблення квадратної матриці.

Завдання. Скласти блок-схему і програму для оброблення багатовимірних масивів. Індивідуальні завдання наведені в таблиці 4.3. Ім'я та розмір матриці вибрати самостійно.

Таблиця 4.3 – Індивідуальні завдання

Вар.	Умова задачі
1	Обчислити добуток негативних елементів головної діагоналі матриці
2	Знайти кількість парних елементів побічної діагоналі матриці
3	Визначити суму негативних елементів головної діагоналі матриці
4	Знайти добуток непарних елементів побічної діагоналі матриці
5	Знайти добуток позитивних елементів побічної діагоналі матриці
6	Знайти суму непарних елементів побічної діагоналі матриці
7	Знайти максимальне негативне число головної діагоналі матриці
8	Знайти суму позитивних непарних елементів головної діагоналі матриці
9	Знайти мінімальне позитивне число побічної діагоналі матриці
10	Знайти суму непарних елементів головної діагоналі матриці
11	Знайти мінімальне парне число головної діагоналі матриці
12	Знайти мінімальне непарне число побічної діагоналі матриці

Продовження таблиці 4.3

Вар.	Умова задачі
13	Знайти суму позитивних кратних 5 елементів побічної діагоналі матриці
14	Знайти кількість негативних не кратних 3 елементів головної діагоналі матриці
15	Знайти кількість позитивних парних елементів головної діагоналі матриці
16	Знайти добуток квадратів негативних елементів головної діагоналі матриці
17	Знайти добуток негативних непарних елементів побічної діагоналі матриці
18	Знайти суму ненульових елементів головної діагоналі матриці
19	Знайти різницю сум негативних і позитивних елементів головної діагоналі матриці
20	Знайти середнє геометричне модулів негативних елементів побічної діагоналі матриці
21	Знайти суму кратних 7 позитивних елементів побічної діагоналі матриці
22	Знайти середнє арифметичне позитивних елементів побічної діагоналі матриці
23	Знайти середнє геометричне кратних 3 елементів головної діагоналі матриці
24	Знайти частку від розподілу кількості негативних елементів побічної діагоналі матриці на їхню суму
25	Знайти добуток ненульових елементів головної діагоналі матриці
26	Знайти середнє арифметичне кратних 5 елементів побічної діагоналі матриці
27	Знайти середнє геометричне ненульових елементів головної діагоналі матриці
28	Знайти добуток кратних 7 елементів побічної діагоналі матриці
29	Знайти частку від розподілу суми негативних елементів головної діагоналі матриці на суму всіх елементів побічної діагоналі
30	Знайти середнє арифметичне негативних елементів побічної діагоналі матриці

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв'язання задачі; програму;

3) рішення задачі (лістинг) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. Як виконується доступ до елементів масиву?

2. Яка кількість операторів циклу необхідна для оброблення головної чи побічної діагоналі масиву?

3. Як використовувати засіб **typedef** для оголошення типу масиву?

4. Який з індексів можна не вказувати при явній ініціалізації масивів?

5. В який послідовності й як відбувається заповнення багатовимірних числових масивів у програмах на мові **C**?

6. Для чого застосовується початкова ініціалізація чисельних масивів при подальшому їх використанні?

7. Скільки потрібно операторів циклу для виведення на консоль двомірного числового масиву?

8. Чому при визначенні розмірності масиву за допомогою препроцессорної директиви *define* не використовується крапка з комою після числового значення?

5 ПОБУДОВА ГРАФІКА ФУНКЦІЇ

Графічні функції призначені для керування відеорежимами роботи дисплея, виведенням графічної інформації на екран.

Принципи програмування на мові С засновані на понятті функції. Наприклад, до системних функцій відносять printf(), scanf(), gets(), putchar() та ін. Функції – це будівельні елементи мови С і те місце, в якому виконується вся робота програми.

Великі програми зазвичай складаються з декількох користувальницьких функцій і ряду системних функцій. Функція – це самостійна одиниця програми. Функції підвищують рівень модульності програми, полегшують її читання, внесення змін і корекцію помилок.

В основі всіх програм на мові програмування С лежать одні і ті ж фундаментальні елементи – функції. Зокрема, функція main() є обов'язковою для будь-якої програми. У всіх програмах С визначається єдина зовнішня функція з ім'ям main(), що служить точкою входу в програму, тобто першою функцією, виконуваною після запуску програми.

Жодна програма на мові С не може обійтися без функцій.

Функція в мові С відіграє ту ж роль, що і підпрограми або процедури в інших мовах. Кожна функція мови має ім'я і список аргументів. За угодою, прийнятою в мові С, при записуванні імені функції після неї ставляться круглі дужки. Ця угода дозволяє легко відрізнити імена змінних від імен функцій.

5.1 Графічні функції

void far detectgraph(int far *graphdriver, int far *graphmode); – визначення доступного відеодрайвера.

void far initgraph(int far * graphdriver, int far *graphmode, char far *pathtodriver); – установлення відеорежиму.

vo**id far setgraphmode(int mode); –** установлення відеорежиму.

void far restorecrtmode(void); – тимчасовий перехід із графічного відеорежиму в текстовий.

void far closegraph(void); – закриття графічної системи.

void far setvisualpage(int page); – установлення активної відеосторінки.

void far setactivepage(void); – виведення на активну відеосторінку.

int far getmaxx(void); – визначення максимального значення координати **x.**

int far getmaxy(void); – визначення максимального значення координати y**.**

void far setviewport(int left, int top, int right, int botton, int clip); – установлення нового графічного вікна.

void far getviewsettings(struct viewporttype far *viewport); – одержання параметрів поточного вікна.

void far moveto(int x, int y); void far moverel(int dx, int dy); – переміщення поточної графічної позиції в координати **x**, **y** або на величину **dx, dy.**

void far setlinestyle(int linestyle, unsigned upattern, int thickness); – установлення типу лінії.

i**nt far getx(void); –** одержання поточної графічної позиції **(x).**

int far gety(void); – одержання поточної графічної позиції **(y).**

void far clearviewport(void); – очищення поточного графічного вікна.

void far cleardevice(void); – очищення активної відеосторінки.

int far getmaxcolor(void); – визначення максимальної кількості кольорів.

void far setpalette(int colornum, int color); – установлення палітри.

void far setbkcolor(int color); – установлення кольору фона.

unsigned far getpixel(int x, int y); – одержання поточних параметрів пікселя**.**

void far putpixel(int x, int y, int color); – виведення пікселя з параметрами.

5.1.1 Графічні примітиви

void far bar(int left, int top, int right, int botton);

void far bar3d(int left, int top, int right, int botton, int depth, int topflag);

void far fillpoly(int numpoints, int far *polypoints);

void far fillelipse(int x, int y, int xradius, int yradius);

void far pielipse(int x, int y, stangle, int endangle, int radius);

void far sector(int x, int y, int stangle, int endangle, int xradius, int yradius);

void far line(int x1, int y1, int x2, int y2);

void far linerel(int dx, int dy);

void far lineto(int x, int y);

void far rectangle(int left, int top, int right, int botton);

void far drawpoly(int numpoints, far *polypoints);

void far circle(int x, int y, int radiuces);

void far arc(int x, int y, int stangle, int endangle, int radius);

void far ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);

Приклад 20. Напишіть програму побудови на екрані дисплея графіка функції $y = \sin(3x)e^{x/3}$. Передбачте можливість запису в текстовий файл графіка цієї функції.

Для розв'язання прикладу використаємо засоби виведення на друк форматованих даних без застосування спеціальних графічних бібліотек.

Результат виконання програми на рисунку 5.1.

Програмний код розв’язання прикладу:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <locale.h>
// Розміри діаграми по ширині і висоті екрана
#define SCREENW  79
#define SCREENH  25
```

```
// Функція побудови графіка заданої функції
void plot (FILE *fout, double a, double b, double (*f) (double))
{
setlocale(LC_ALL, «Russian»);
// Формальні параметри функції plot
// FILE *fout – вказівник на поток
// double a – ліва межа осі абсцис
// double b – права межа осі абсцис
// double (*f) (double) – вказівник на функцію
        char screen[SCREENW][SCREENH];
        double x, y[SCREENW];
        double ymin = 0, ymax = 0;
        double hx, hy;
        int i, j;
        int xz, yz;
// hx – крок по осі абсцис
        hx = (b - a) / (SCREENW - 1);
        for (i = 0, x = a; i < SCREENW; ++i, x += hx) {
// визначаємо значення функції
        y[i] = f (x);
// запам'ятаємо мінімальне та максимальне значення
        if (y[i] < ymin) ymin = y[i];
        if (y[i] > ymax) ymax = y[i];
        }
        hy = (ymax - ymin) / (SCREENH - 1);
        yz = (int)floor (ymax / hy + 0.5);
        xz = (int)floor (-a / hx + 0.5);
// малювання осей координат
        for (j = 0; j < SCREENH; ++j) {
                for (i = 0; i < SCREENW; ++i) {
```

```
                        if (j == yz && i == xz)
                                screen[i][j] = '+';
                        else if (j == yz)
                                screen[i][j] = '-';
                        else if (i == xz)
                                screen[i][j] = '|';
                        else
                                screen[i][j] = ' ';
                }
        }
// малювання графіка функції
        for (i = 0; i < SCREENW; ++i) {
                j = (int)floor ((ymax - y[i]) / hy + 0.5);
   screen[i][j] = '.'; // символ побудови графіка
        }
// вивід результату у файл або у стандартний потік stdout
        for (j = 0; j < SCREENH; ++j) {
                for (i = 0; i < SCREENW; ++i)
                        fputc (screen[i][j], fout);
                fprintf (fout, «\n»);
        }
}
// Задана функція
double f (double x)
{
        return sin (3.0*x) * exp (-x / 3.0);
}
int main (void)
{
// Вивід графіка у стандартний потік (консоль)
```

```
    plot (stdout, 0.0, 10.0, f);
printf("\n\n Натисніть будь-яку клавішу: «);
    _getch();
    return 0;}
```

У програмі використовується покажчик на файл, який може бути стандартним потоком, тобто екран дисплея. У головної функції *main()* відбувається звернення до функції побудови графіка *plot*(), до якої вводять фактичні параметри, зокрема файл – це *stdout*, тобто стандартний потік, 0.0 – це ліва межа осі абсцис, 10.0 – права межа осі абсцис, *f* – ім'я функції з описом залежності *y = f(x).*

Приклад виконання програми показаний на рисунку 5.1.

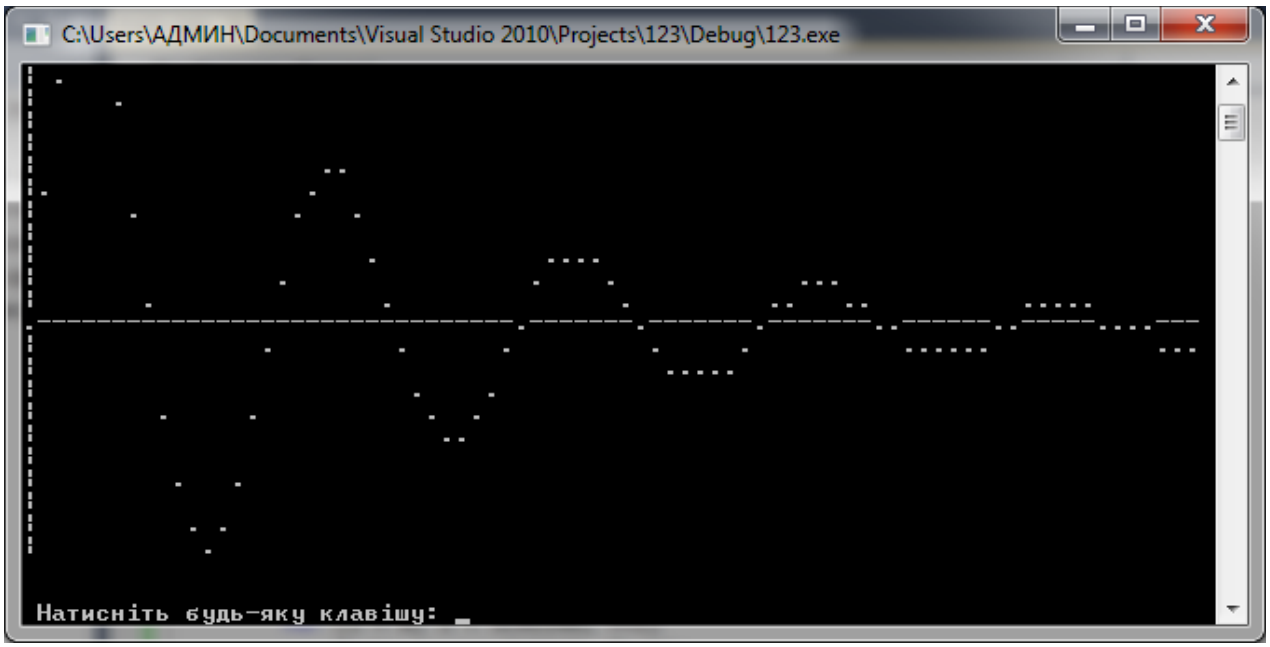


Рисунок 5.1 – Приклад побудови графіка функції на консолі

5.2 Лабораторна робота № 13. Побудова графіка функцій

***Мета роботи*:** навчитися будувати графік функції на мові програмування Сі.

Завдання

Згідно зі своїм варіантом (таблиця 5.1) створити програму для побудови графіка функції.

Таблиця 5.1 – Індивідуальні завдання

Варіант	**Функція для побудови графіка**
1	$\sin^3 2x$
2	$2\sqrt{x^3}\sin x^3$
3	$\sqrt[3]{x^3}\sin x$
4	$5x^2\sin^3 x^3 + 2x^3$
5	$x^3\cos^2\left(x^5 + 2x\right)$
6	$3x^3\sin^2 x^5$
7	$\ln(\sin 4x + 1)^2$
8	$\sin^3 x^2$
9	$\cos^3\left(x^3 + 2\right)^2$
10	$x^2\cos x$
11	$\sin\left(\sqrt{x^5 + 2x}\right)$
12	$(x-1)^3 + \cos 2x^3$
13	$\cos\left(x^5 + 2x^3 - x\right)^5$
14	$x\sin\left(x^2 + 2x\right)$

Продовження таблиці 5.1

Варіант	Функція для побудови графіка
15	$\ln(\cos 2x+1)^3$
16	$5^{x+1}\sin\left(x^3+1\right)$
17	$3\ln\sqrt[5]{\sin x+x^2}$
18	$\sin x^{2x}+\cos\left(x^2+2\right)$
19	$x^x-\cos x$
20	$x^2+\sin 5x$
21	$\sqrt{x^3-1}+\sin x^2$
22	$2\sin(x-e^{-x})$
23	$\sin x^2+x^{0,25}$
24	$\sqrt[3]{\sin^2 x+\cos^4 x}$
25	$\sin^3 x^4$
26	$\sin\sqrt[3]{x^3+x^2}$
27	$x^x\cos x$
28	$x^4\sin 4x$
29	$\sin^2 x^3$
30	$(x+1)^2\cos x^3$

Звіт повинен містити:

1) мету роботи; умову задачі;

2) блок-схему алгоритму розв'язання задачі; програму;

3) рішення задачі (графік) на ЕОМ;

4) короткі висновки з роботи.

Контрольні питання

1. Яка функція застосовується для установки відеорежиму, ініціалізації графічного режиму роботи?
2. Що означають параметри функцій у приведеному прикладі?
3. Як закрити графічний режим роботи?
4. Чи можна одержати й установити координати курсору на екрані ? У чому виміряються ці координати?
5. Які функції дозволяють установлювати колір для виведеної інформації, колір фону, здійснювати різні види заливання зображення?
6. Які графічні примітиви можна зобразити за допомогою бібліотечних функцій?

СПИСОК ЛІТЕРАТУРИ

1. **Вінник, В. Ю.** Алгоритмічні мови та основи програмування: мова С / Вінник В. Ю. – Житомир : ЖДТУ, 2007 – 328 с. – ISBN 978-966-683-143-2.

2. **Дейтел, П.** Как программировать на C++. Введение в объектно-ориентированное проектирование с использованием UML / П. Дейтел. – М. : БИНОМ, 2002. – 453 с.

3. **Джамса, К.** Учимся программировать на языке C++ : пер. с англ. / К. Джамса. – М. : Мир, 1997. – 320 с.

4. Інформаційні технології у медицині : навч. посіб. для студ. вищ. мед. навч. закл. / В. І. Федів, В. Ф Мислицький, К. Б. Тимочко, В. Ф. Боєчко, М. В. Шаплавський. – Чернівці, 2004. – 242 с.

5. **Коплиен, Дж.** Программирование на C++ / Дж. Коплиен. – СПб. : ПИТЕР, 2005. – 624 с.

6. Медична інформатика : підручник для студентів медичних ВНЗ / за ред. В. Г. Кнігавка. – Харків : ХНМУ, 2015. – 240 с.

7. **Павловская, Т. А.** C/C++. Структурное и объектно-ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. СПб. : Питер, 2010. – 329 с.

8. Практикум для виконання лабораторних робіт з дисципліни «Мови об’єктно-орієнтованого програмування» / укл. : О. Ф. Тарасов, О. В. Алтухов. – Краматорськ : ДДМА, 2001. – 152 с.

9. Програмування на мові С++ у середовищі Visual Studio 2010 : навчальний посібник для студентів спеціальності 141 «Електроенергетика, електротехніка та електромеханіка» / С. Л. Загребельний, С. В. Малигіна, М. В. Брус, С. С. Гурковська. – Краматорськ : ДДМА, 2019. – 146 с.

10. **Скляров, В. А.** Язык С++ и объектно-ориентированное программирование: справочное издание / В. А. Скляров. – Минск : Вышэйшая школа, 1997. – 480 с.

11. С++. Основи програмування. Теорія та практика : підручник / [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : Фенікс, 2010. – 544 с.

12. **Шпак, З. Я.** – Програмування мовою С : навчальний посібник / Шпак З. Я. – Львів : Оріяна – Нова. 2006.

13. Программирование на языке С++ в Microsoft Visual Studio 2010. – [Режим доступу]: https://www.intuit.ru/studies/courses/495 -/351/lecture/8377

14. Handbook of Medical Informatics. Editors: J. H. van Bemmel, M. A. Musen. – http://www.mieur.nl/mihandbook; http://www.mihandbook.stanford.edu.

15. **Stanley, B**. **C++** Primer: Fifth Edition / Stanley B., Lippman, Josée Lajoie, Barbara E. Moo – Addison Wesley, 2013.

16. **Thomas Cormen**. Introduction to Algorithms Third Edition / Thomas Cormen, Charles Leiserson, Ronald Rivest, Clifford Stein – MIT Press, 2009.

Навчальне видання

ВАСИЛЬЄВА Людмила Володимирівна,
МАЛИГІНА Світлана Валеріївна,
БЕРЕЖНА Олена Валеріївна

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ, ОБРОБЛЕННЯ МЕДИЧНИХ ДАНИХ

Навчальний посібник

для здобувачів вищої освіти
спеціальності 122 «Комп'ютерні науки»

Редагування, комп'ютерне верстання Я. О. Бершацька

/2022. Формат 60 × 84/16. Ум. друк. арк. 5.99.
Обл.-вид. арк. 3.27. Тираж 50 пр. Зам. № 6

Видавець і виготівник
Донбаська державна машинобудівна академія
84313, м. Краматорськ, вул. Академічна, 72.
Свідоцтво суб'єкта видавничої справи
ДК №1633 від 24.12.03