

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ІНЖЕНЕРНО-ТЕХНІЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

**МЕТОДИЧНІ ВКАЗІВКИ І ЗАВДАННЯ ДО ЛАБОРАТОРНИХ РОБІТ
З КУРСУ «СТРУКТУРИ ДАНИХ ТА АЛГОРИТМИ»
для студентів 1-го курсу
інженерно-технічного факультету
спеціальності 123 Комп'ютерна інженерія»**

Ужгород – 2019

Методичні вказівки і завдання до лабораторних робіт з курсу «Структури даних та алгоритми» для студентів 1-го курсу інженерно-технічного факультету спеціальності 123 Комп'ютерна інженерія.

Укладачі: – кандидат фізико-математичних наук, доцент кафедри комп'ютерних систем та мереж Горват П.П., старший викладач кафедри комп'ютерних систем та мереж Самусь Є.І., викладач кафедри комп'ютерних систем та мереж Мишковська Х. М.

Рецензент: Мулеса О. Ю. – канд.техн. наук, доцент кафедри кібернетики та прикладної математики. УжНУ.

Відповідальний за випуск – Туряниця І.І., канд. фіз.-мат. наук, доцент, декан інженерно-технічного факультету.

Дані методичні вказівки розглянуто та схвалено на засіданні кафедри комп'ютерних систем та мереж, протокол № 8 від 26 лютого 2019 року.

ВСТУП

Метою виконання лабораторних робіт є формування практичних вмінь та навичок створення та опрацювання статичних, напівстатичних та динамічних, лінійних та нелінійних структур даних, а також використання типових алгоритмів на цих структурах.

В результаті виконання лабораторних робіт студенти повинні знати основні статичні, напівстатичні та динамічні структури даних, лінійні та нелінійні структури даних, а саме масиви структур, списки різних видів, стеки, черги, дерева та графи; основні алгоритми опрацювання типових структур даних, алгоритми створення/видалення структури, вставки/видалення елемента структури, очищення структури і т.п.; алгоритми впорядкування та пошуку.

Порядок виконання лабораторних робіт. Для виконання лабораторних робіт необхідно:

1. використовуючи літературні джерела, конспект лекцій, методичні розробки з дисципліни, засвоїти теоретичний матеріал, пов'язаний з тематикою лабораторної роботи;
2. відповісти викладачу, що забезпечує проведення лабораторних занять на поставлені запитання за темою лабораторної роботи;
3. опрацювати наведений у лабораторній роботі приклад;
4. отримати індивідуальне завдання;
5. розробити схему алгоритму для розв'язування індивідуального завдання;
6. написати відповідну програмну реалізацію;
7. протестувати програму для різних вхідних даних;
8. виконати програму та зафіксувати отримані результати згідно завдання;
9. перевірити правильність роботи програми; при необхідності внести зміни у програму та зафіксувати остаточні результати;
10. оформити та захистити звіт про виконання лабораторної роботи.

Лабораторна робота № 1

Основні теоретичні поняття

1. Поняття алгоритму

Алгоритм — це послідовність дій над заданими об'єктами, чітко та однозначно визначаюча обчислювальний процес.

Ефективним методом побудови алгоритмів є метод покрокової деталізації, при якому завдання розбивається на кілька простих підзадач (модулів), і для кожного модуля створюється свій власний алгоритм.

Здебільшого модуль реалізує певний процес обробки інформації і застосовується як для окремого використання, так і для включення модуля в інші алгоритми. Застосування модульності при створенні алгоритмів дозволяє розбити великі завдання на незалежні блоки (модулі), усуває повторення стандартних дій і значно прискорює процес відлагодження алгоритму в цілому.

Найчастіше головний модуль алгоритму містить декілька інших модулів, створених раніше. Використовуючи модулі як складові великої конструкції, можна створювати алгоритми будь-якого ступеня складності, і при цьому не втрачати контролю за функціонуванням алгоритму всієї задачі.

Такий метод називається структурним проектуванням алгоритму, він є універсальним і може використовуватися як для обчислювальних процесів (так зване системне програмування), так і для процесів реального життя.

Властивості алгоритму

1. **Дискретність** – процес розв'язку розбивається на кроки. Кожен крок – це одна дія або підпорядкований алгоритм. Таким чином полегшується процес знайдення помилок і редагування алгоритму.
2. **Визначеність (точність)** – кожен крок алгоритму має бути однозначно описаною дією і не містити двозначностей.
3. **Зрозумілість** – усі дії, включені до алгоритму, мають бути у межах компетенції виконавця алгоритму.
4. **Універсальність (масовість)** – алгоритм має виконуватися при будь-яких значеннях вхідних даних та початкових умов.
5. **Скінченність** – алгоритм має бути реалізований за кінцеве число кроків і повинен використовувати кінцевий набір вхідних значень.
6. **Результативність** – алгоритм має привести до отримання результату.

Способи подання алгоритмів

Алгоритми можуть бути подані

- словесно (засобами природної мови у вигляді плану дій)
- графічно (у вигляді блок-схем)
- у вигляді програм, написаних певною мовою програмування.

Найчастіше алгоритми обчислювальних процесів подаються у вигляді блок-схем, де кожний крок алгоритму представлено спеціальним блоком, який показує дію, яку треба виконати.

Графічному опису передуює, як правило, побудова математичної моделі – математичного опису алгоритму. Такий опис полягає у формалізованому (із застосуванням математичних символів) поданні всіх розглянутих залежностей і методів відшукування значень вихідних даних на підставі вхідних.

Призначення блоків впливає з їхніх назв. Блоки поєднуються між собою лініями потоку. Природні напрями потоків зверху вниз і зліва направо. Якщо напрямок потоку інший то лінія повинна мати стрілку.

Рекомендується не перетинати лінії потоку, а використовувати поєднувач блоків. Для використання поєднувача блоки мають бути попередньо пронумеровані, а сам поєднувач має містити цифру – номер блоку, з яким відбувається поєднання або номер блоку, з якого відбувається поєднання. Інакше: при з'єднуванні блоків можливо використовувати спільний символ, який записується в поєднувач блоків.

Основні види блоків та їх призначення наведені у таблиці 1.1.

2. АЛГОРИТМИ ЛІНІЙНОЇ СТРУКТУРИ

Лінійна структура використовується в алгоритмах, де одна дія виконується слідом за іншою послідовно і при цьому жодна з дій не пропускається і не повторюється.

Розглянемо приклади алгоритмів лінійної структури.

Приклад 2.1. Обчислити висоти трикутника зі сторонами **a**, **b**, **c** за формулами

$$h_a = \frac{2}{a} \sqrt{p(p-a)(p-b)(p-c)}$$
$$h_b = \frac{2}{b} \sqrt{p(p-a)(p-b)(p-c)}$$
$$h_c = \frac{2}{c} \sqrt{p(p-a)(p-b)(p-c)},$$

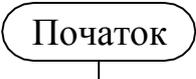
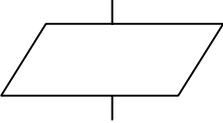
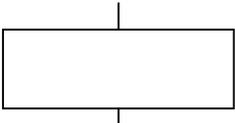
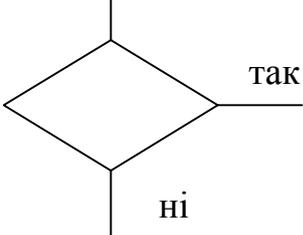
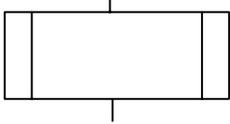
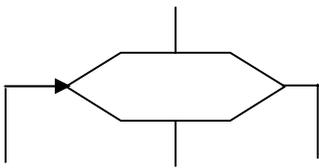
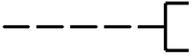
$$\text{де } p = \frac{a + b + c}{2}$$

Введемо позначення:

$$t = 2\sqrt{p(p-a)(p-b)(p-c)} \Rightarrow h_a = \frac{t}{a}, \quad h_b = \frac{t}{b}, \quad h_c = \frac{t}{c}$$

Блок-схема алгоритму наведена на рисунку 2.1.

Основні види блоків, їх призначення

Графічне зображення	Назва, виконувана дія
	Початок алгоритму
	Закінчення алгоритму
	Блок вводу-виводу
	Виконання обчислень або присвоєння значень
	Перевірка умови. Якщо умова справедлива (набуває значення ІСТИНА), виконується перехід по лінії Так , а якщо не справедлива (набуває значення БРЕХНЯ), то виконується перехід по лінії Ні
	Виклик раніше створених алгоритмів (модулів)
	Блок організації циклу
	Коментар. Короткі пояснення до показаного блоку
	Поєднувач блоків

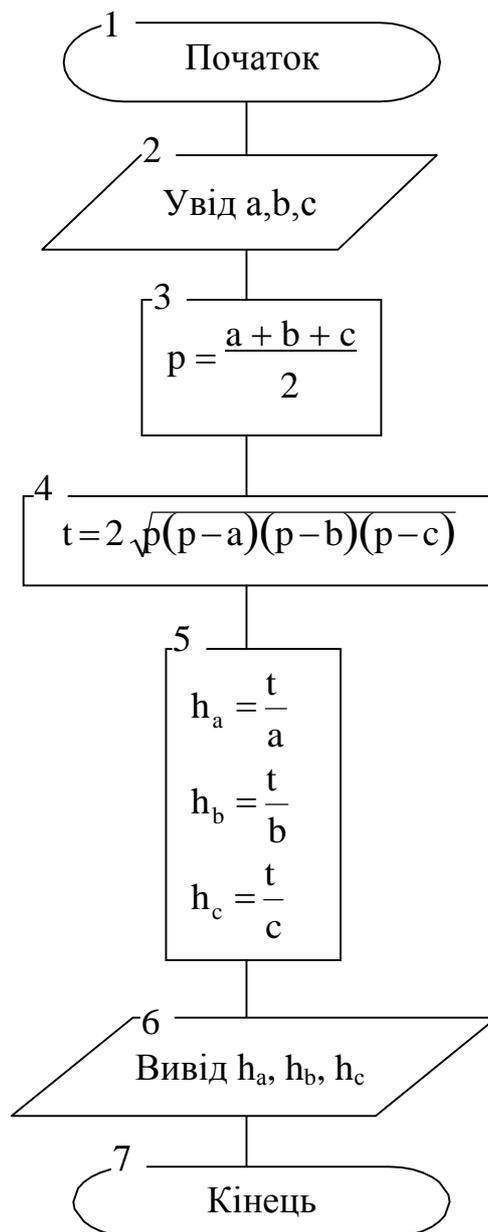


Рис. 2.1. Блок-схема алгоритму прикладу 2.1.

Приклад 2.2. Для заданих значень **a** і **c** знайти значення виразу:

$$Y = \sqrt{\left| x^2 + c \right|} - \sin(ax - c),$$

де $x = \cos(a^2 c^2)$

Словесний спосіб надання алгоритму:

1. Уведення вхідних значень змінних **a**, **c**.
2. Обчислення виразу $x = \cos(a^2 c^2)$.
3. Обчислення виразу $Y = \sqrt{\left| x^2 + c \right|} - \sin(ax - c)$.
4. Виведення вихідних значень **x** та **Y**.

На рисунку 2.2. подана блок-схема алгоритму.

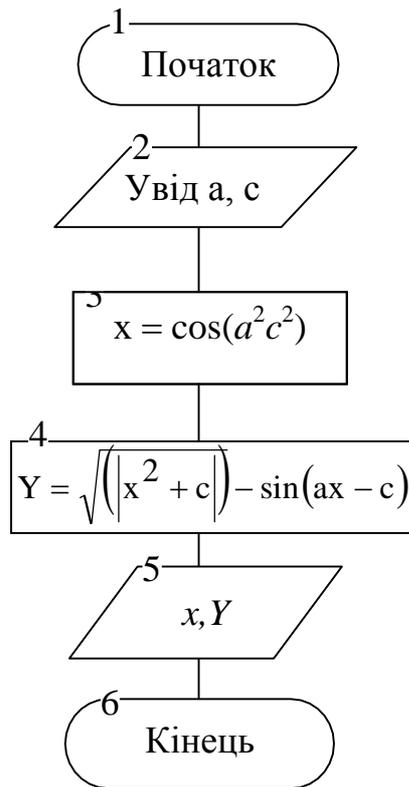


Рис. 2.2. Блок-схема алгоритму прикладу 2.2.

3. АЛГОРИТМИ РОЗГАЛУЖЕНОЇ СТРУКТУРИ

Розгалужена структура передбачає вибір виконання дії залежно від виконання певної умови, при цьому деякі дії можуть не виконуватися взагалі (пропускатися).

Проста умова містить два вирази (значення), поєднані знаком операції відношення:

- > більше за...
- < менше за...
- ≥ більше або дорівнює...
- ≤ менше або дорівнює ...
- ≠ не дорівнює...

Результатом перевірки умови є логічний вираз ІСТИНА, якщо умова виконується, або БРЕХНЯ, якщо умова не виконується.

Приклад 3.1. Знайти значення дійсних коренів квадратного рівняння

$$ax^2 + bx + c = 0, \quad a \neq 0$$

Словесний спосіб подання алгоритму:

1. Уведення значень коефіцієнтів **a, b, c**.
2. Обчислення значення дискримінанта за формулою $D = b^2 - 4ac$.
3. Перевірка отриманого значення дискримінанта: якщо дискримінант ≥ 0 , то перехід на п.4, в противному разі перехід на п.6.
4. Обчислення дійсних коренів рівняння за формулами:

$$x_1 = \frac{-b + \sqrt{D}}{2a}$$

$$x_2 = \frac{-b - \sqrt{D}}{2a}.$$

5. Виведення отриманих результатів x_1 та x_2 . Кінець алгоритму.
6. Виведення повідомлення «Дійсних коренів немає». Кінець алгоритму.

На рисунку 3.1. наведена блок-схема алгоритму:

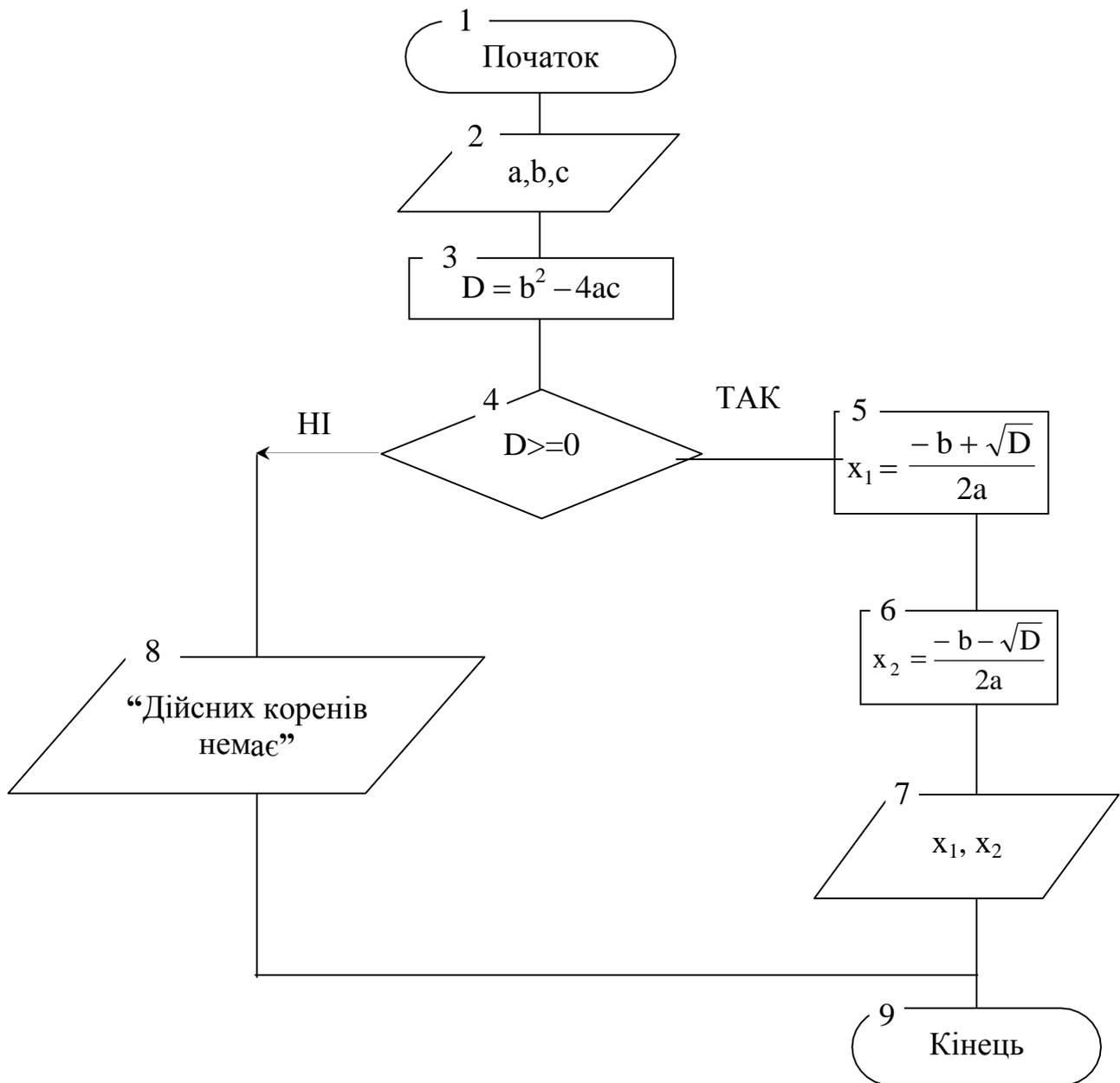


Рис. 3.1. Блок-схема алгоритму прикладу 3.1.

У блок-схемі алгоритму блок №4 використовується для перевірки умови і реалізує розгалуження: якщо умова набуває результату **істина**, то алгоритм продовжується по блоках 5, 6, 7, 9, а блок 8 зовсім не виконується. Навпаки,

коли умова набуває значення **брехня**, алгоритм продовжується по блоках 8, 9, при цьому блоки 5, 6, 7 не виконуються.

Приклад 3.2. Для заданих значень x, a, b обчислити значення виразу:

$$Y = \begin{cases} \frac{abx^2}{b+x}, & x \leq 2 \\ a + \sin(bx), & 2 < x \leq 8 \\ \sqrt{|ab+x|}, & x > 8 \end{cases}$$

На рисунку 3.2. наведена блок-схема алгоритму.

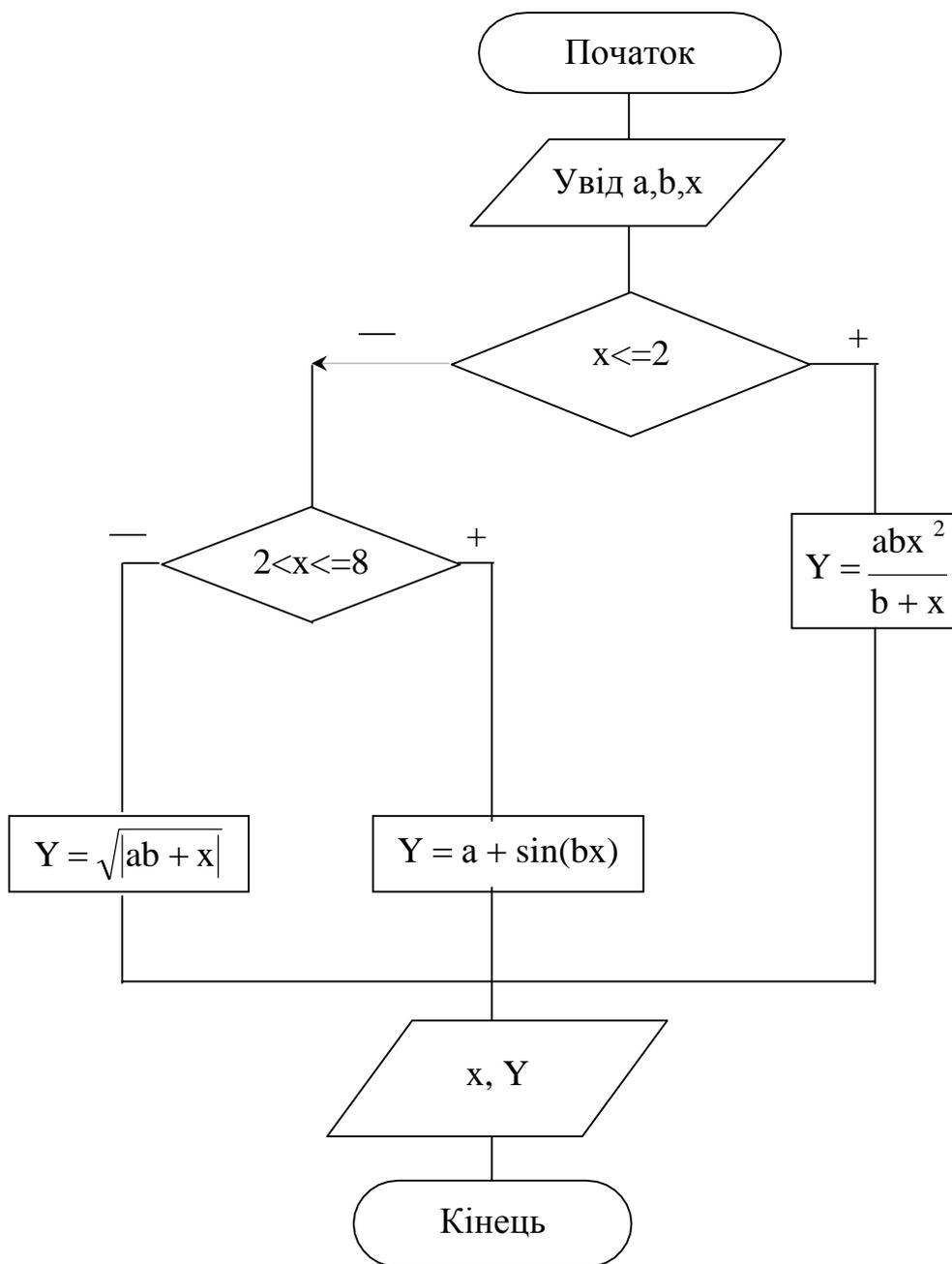


Рис. 3.2. Блок-схема алгоритму прикладу 3.2.

4. АЛГОРИТМИ ЦИКЛІЧНОЇ СТРУКТУРИ

Циклом називають частину алгоритму, яка повторюється.

При кожному черговому виконанні циклу перевіряється умова на продовження роботи і, якщо умова набуває результату ІСТИНА, цикл виконується, а якщо умова набуває результату БРЕХНЯ – цикл не виконується.

Перевірка умови може бути організована на початку циклу, і такий цикл називається циклом з передумовою, або у кінці циклу – тоді такий цикл називається циклом з післяумовою.

Різниця між такими циклами полягає в тому, що цикл з післяумовою виконується хоча б один раз, а цикл з передумовою може не виконуватися жодного разу.

Цикл по лічильнику характерний тим, що заздалегідь відома кількість повторень циклу, і цикл буде виконуватися, доки значення лічильника циклу не перевищить зазначену кількість повторень.

Якщо відомі початкове та кінцеве значення параметра циклу, а також закон (формула), за яким це значення змінюється, то цикл буде виконуватися, доки параметр циклу лежатиме у межах від початкового до кінцевого значення.

Приклад 4.1. Побудувати таблицю значень функції

$$Y = a + \sin(bx)$$

для заданих коефіцієнтів **a** і **b** та аргументу **x**, що змінюється від **-4** до **6** з кроком **2**.

Словесний спосіб подання алгоритму:

1. Уводяться коефіцієнти **a** і **b**.
2. Задається початкове значення аргументу **x = -4**.
3. Обчислюється значення функції **Y** для поточного аргументу.
4. Виводиться отримане значення функції **Y**.
5. Значення аргументу **x** збільшується на **2**.
6. Перевіряється умова продовження циклу: якщо нове значення аргументу не перевищує заданого кінцевого значення **6**, то цикл (пункти – б) виконується ще раз, у протилежному випадку — кінець алгоритму.

На рисунку 4.1. подана блок-схема алгоритму.

Цей цикл є циклом з післяумовою, тому що перевірка умови проводиться у кінці циклу.

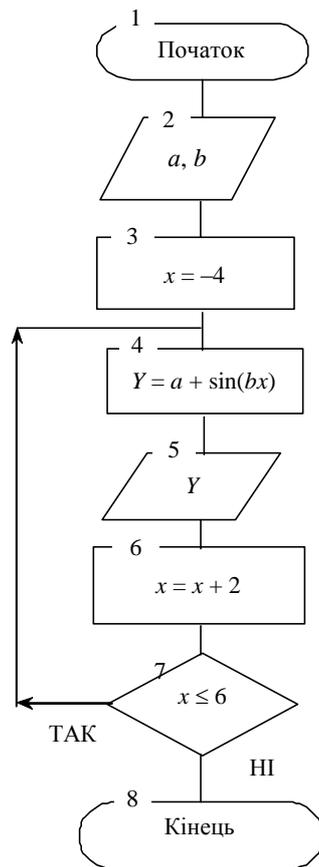


Рис. 4.1. Блок-схема прикладу 4.1.

Приклад 4.2. Обчислити значення функції

$$Y = \frac{b^2}{x^2 + b},$$

де x змінюється від $x_{\text{початкове}} = 0$ до $x_{\text{кінцеве}} = 3$
з кроком $\Delta x = 0,1$; $b = 3,8$

Число повторень циклу n обчислюється за формулою

$$n = \left[\frac{x_k - x_n}{\Delta \zeta} \right] + 1$$

У нашому випадку цикл буде працювати 31 раз, тобто отримаємо 31 значення функції Y . Блок-схема алгоритму може бути наведена двома способами (рисунок 4.2. а, б)

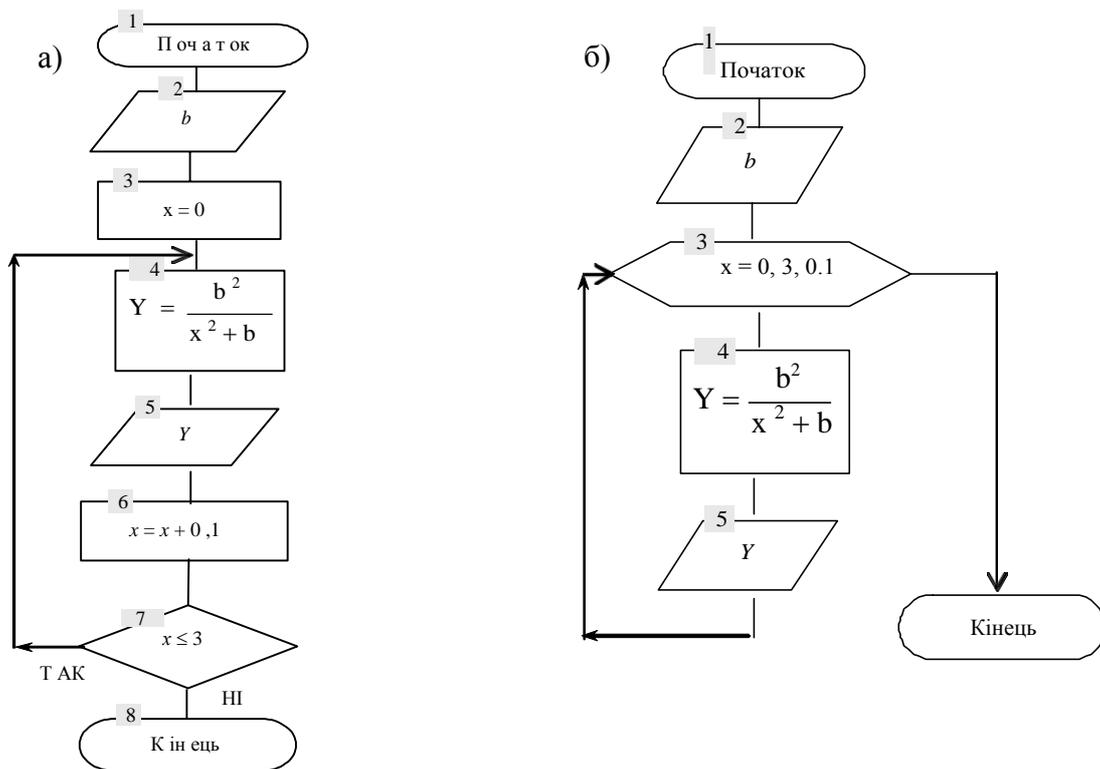


Рис. 4.2. Блок-схеми прикладу 4.2.

Приклад 4.3. Обчислити значення функції

$$Y = \begin{cases} \alpha \sin x, & x \leq 0,2 \\ \cos x + 1, & x > 0,2 \end{cases}$$

де $x \in [-5; 5]$, $\Delta x = 0.2$, $\alpha = 5x$

Блок-схема алгоритму наведена на рисунку 4.3.

5. ІТЕРАЦІЙНИЙ ЦИКЛ

Ітераційний цикл характерний тим, що виконується, доки проміжне значення не досягне зазначеної величини, тобто кількість повторень циклу заздалегідь невідома.

Приклад 5.1. Обчислити значення членів нескінченного ряду

$$x; \frac{x^2}{2!}; \frac{x^3}{3!}; \dots; \frac{x^n}{n!}, \dots \quad \text{де } x \text{ — задане число}$$

Обчислення проводити, доки не виконається умова $\frac{x^n}{n!} \leq \varepsilon$ ⁽¹⁾, де ε – задане значення (точність обчислення).

У даному випадку ми не знаємо, при якому значенні n виконається умова (1).

Число повторень циклу залежить від проміжного результату, де:

$y_1 = x$ – перший член ряду;

$y_2 = y_1 \cdot \frac{x}{2}$ – другий член ряду;

$y_3 = y_2 \cdot \frac{x}{3}$ – третій член ряду;

.....

$y_n = y_{n-1} \cdot \frac{x}{n}$ – рекурентна формула для n -го члена ряду

Для того, щоб використати цю формулу для 1-го члена ряду $y_1 = y_0 \cdot \frac{x}{1}$

необхідно, щоб $y_0 = 1$.

На рисунку 5.1. подана блок-схема алгоритму.

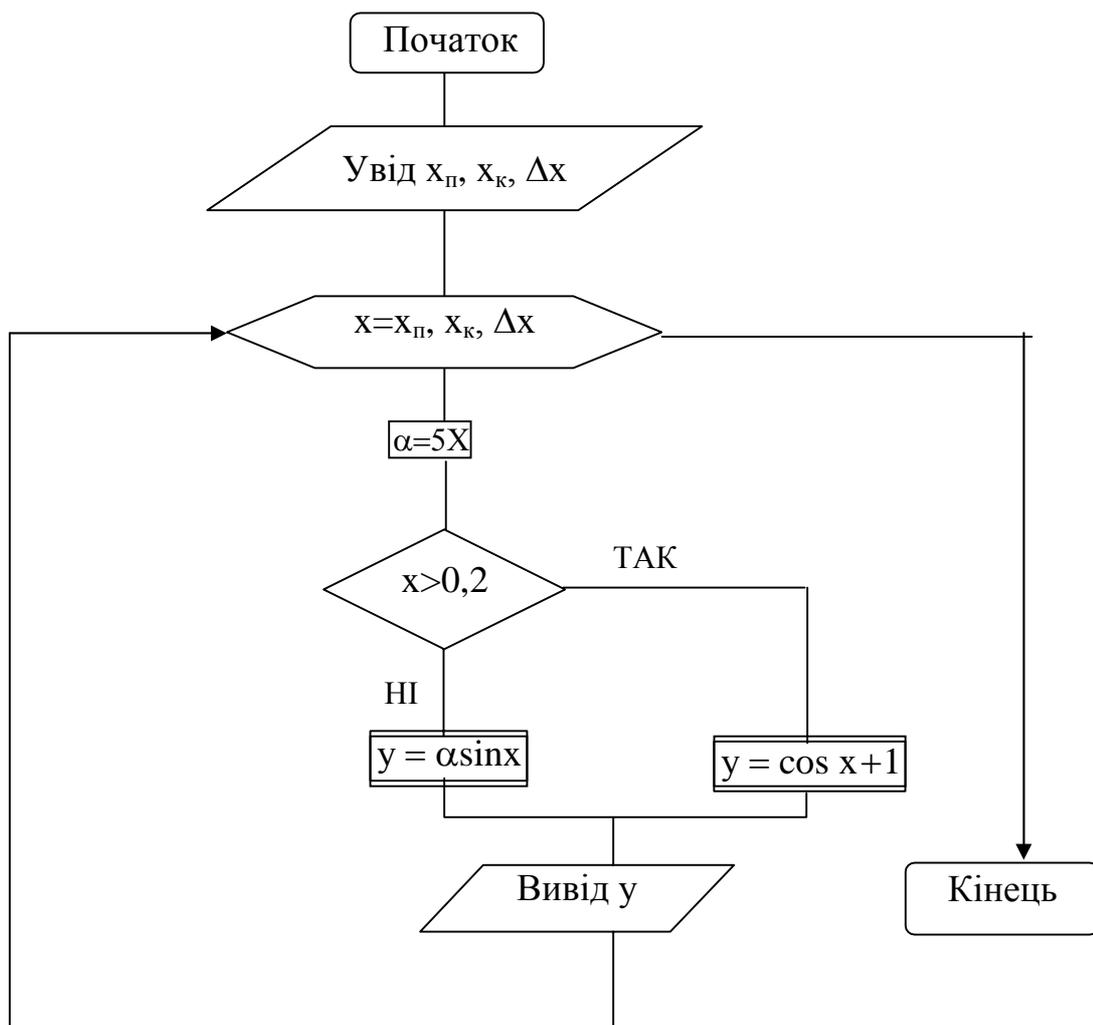


Рис. 4.3. Блок-схема прикладу 4.3.

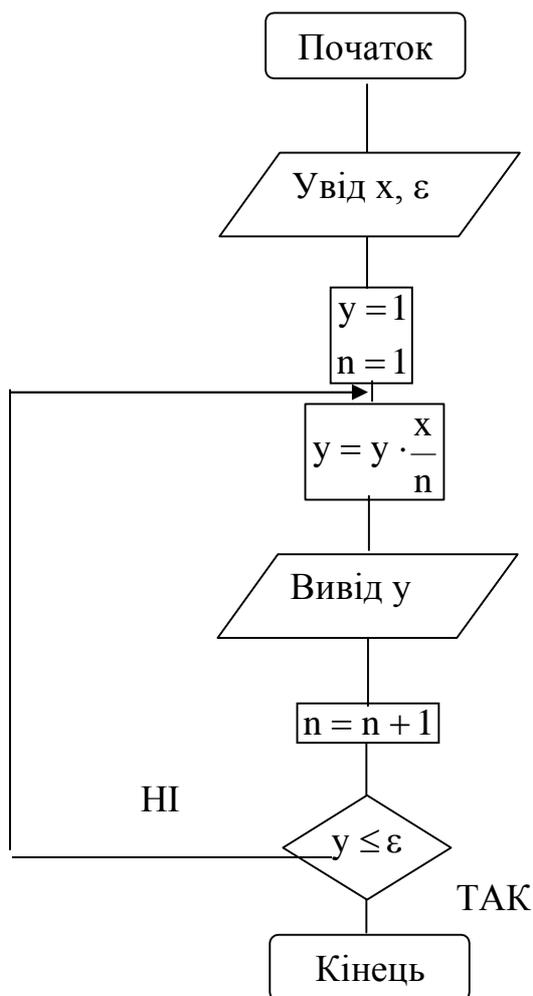


Рис. 5.1. Блок-схема прикладу 5.1.

Приклад 5.2. Обчислити вираз

$$Y = \sqrt[n]{100}, \quad \text{де } n = 2, 3, \dots$$

Визначити таке n , при якому $Y \leq \epsilon$,
якщо σ – задане значення (точність обчислення).

Словесний спосіб надання алгоритму:

1. Задається початкове значення $n = 2$.
2. Обчислюється функція Y .
3. Проводиться перевірка, чи отримане значення функції дорівнює або менше за σ .
4. Якщо результат перевірки умови ІСТИНА, то виводиться поточне значення n та отримане значення функції, цикл припиняється. Кінець алгоритму.
5. Якщо результат перевірки умови БРЕХНЯ, значення n збільшується на одиницю ($n + 1$), і цикл повторюється з п.2.

На рисунку 5.2. подана блок-схема алгоритму.

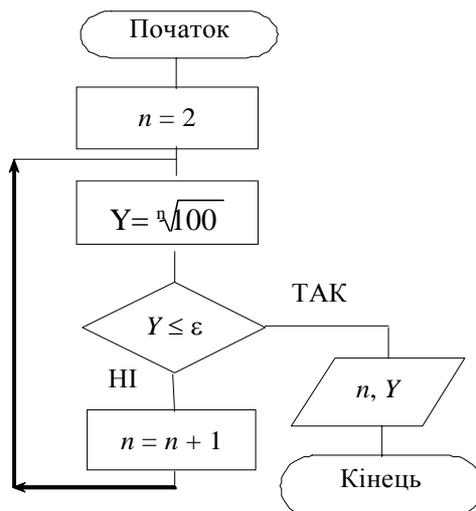


Рис. 5.2. Блок-схема прикладу 5.2

Приклад 5.3. Задано натуральне число N. Визначити кількість цифр у ньому. Блок-схема алгоритму наведена на рисунку 5.3.

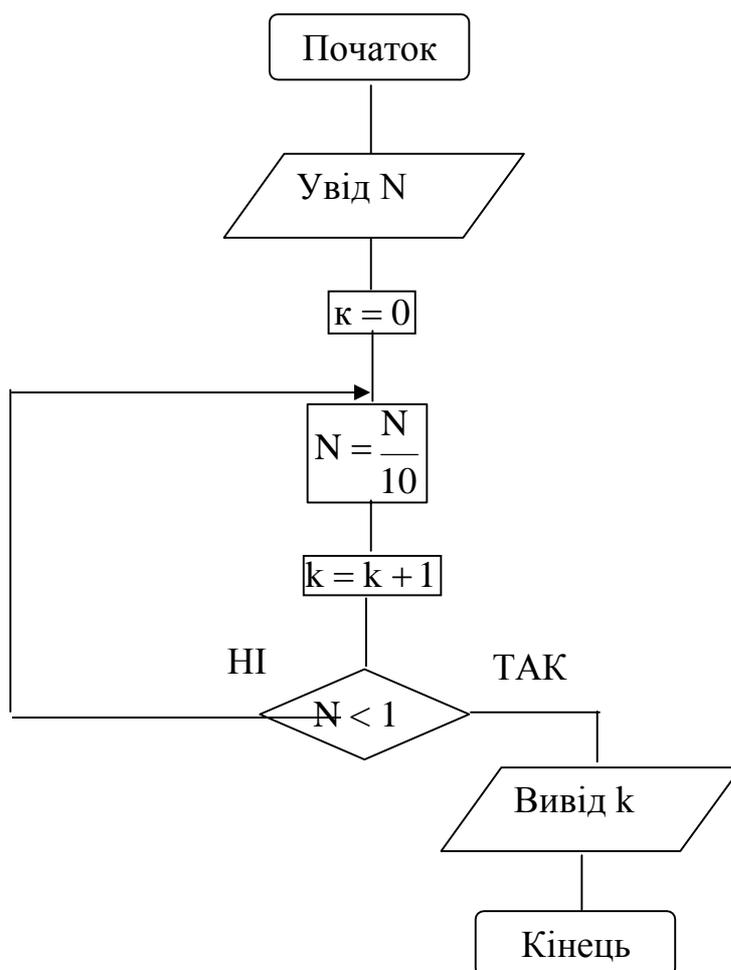


Рис. 5.3. Блок-схема прикладу 5.3.

Задача вирішується шляхом ділення числа на 10. Цикл буде працювати до тих пір, поки число не стане менше 1. K – кількість цифр у числі.

6. ОДНОМІРНІ МАСИВИ

6.1. Табулювання функції

Приклад 6.1. Побудувати таблицю значень функції

$$y = \frac{b^2}{x^2 + b},$$

де X – масив чисел, $X = (x_1, x_2, \dots, x_n)$, $b = 4,8$.

На рисунку 6.1. зображено блок-схему алгоритма.

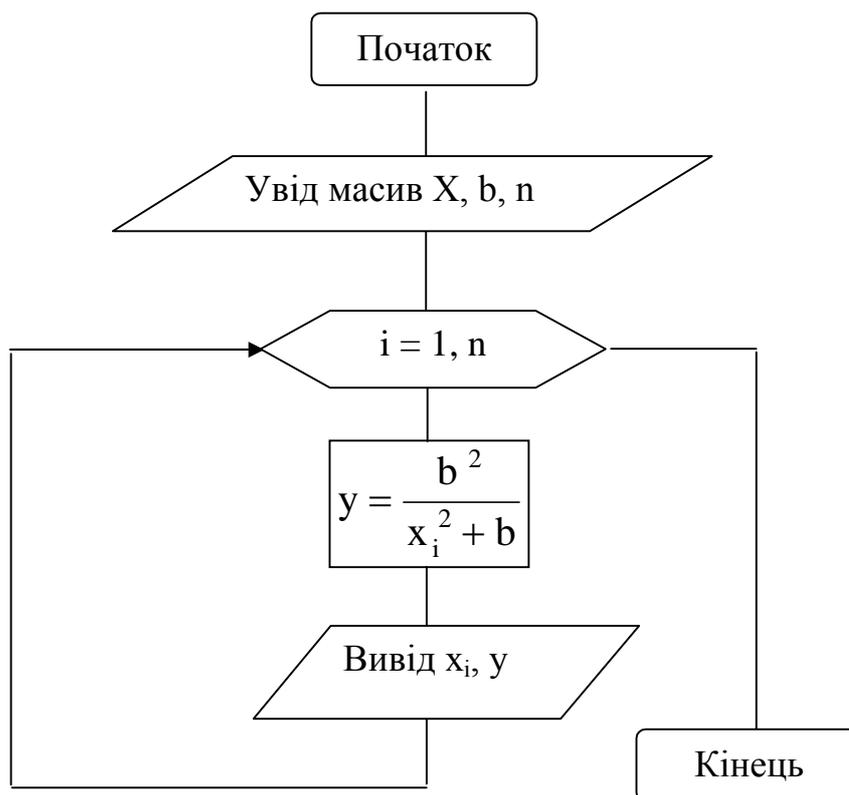


Рис. 6.1. Блок-схема прикладу 6.1

Нехай $X = (3; -0,8; 16,4; 100; -17; 2,4)$.

У даному випадку цикл буде працювати 6 разів, i – параметр циклу.

6.2. Накопичення суми і добутку елементів масиву

Приклад 6.2.1. Обчислити суму елементів масиву $\alpha = (\alpha_1, \alpha_2 \dots \alpha_n)$.

$$S = \sum_{i=1}^n \alpha_i = \alpha_1 + \alpha_2 + \dots + \alpha_n \quad (\text{рис. 6.2.1.})$$

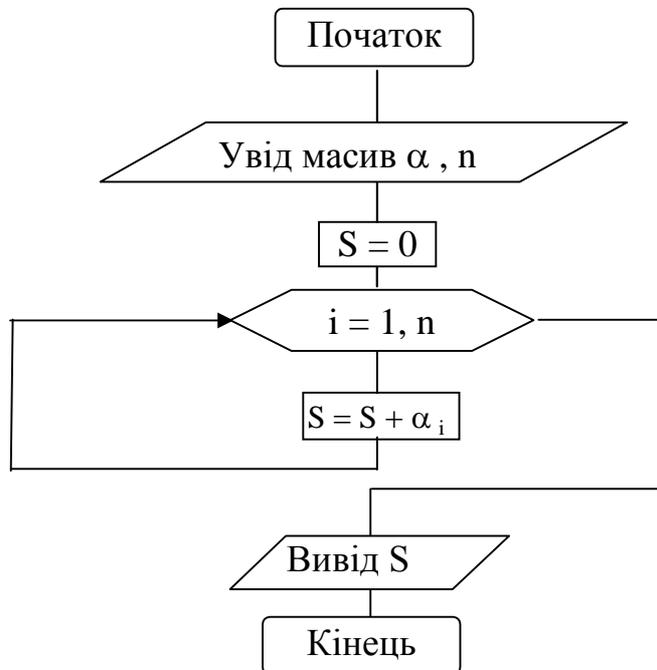


Рис. 6.2.1. Блок-схема алгоритма прикладу 6.2.1.

Приклад 6.2.2. Обчислити добуток елементів масиву $\alpha = (\alpha_1, \alpha_2 \dots \alpha_n)$.

$$P = \prod_{i=1}^n \alpha_i = \alpha_1 \cdot \alpha_2 \cdot \dots \cdot \alpha_n \quad (\text{Рис. 6.2.2.})$$

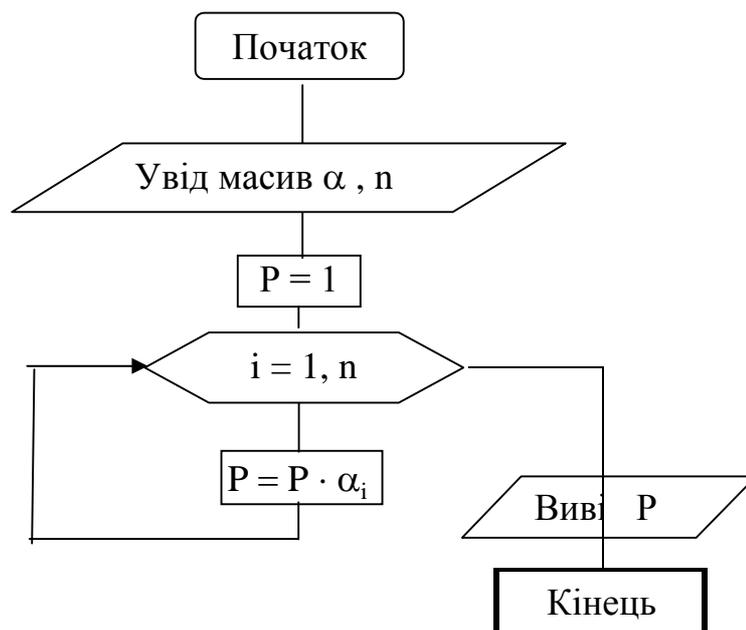


Рис. 6.2.2. Блок-схема алгоритма прикладу 6.2.2

6.3. Пошук максимального і мінімального елементів масиву

Приклад 6.3. Дан масив чисел $X = (x_1, x_2 \dots x_n)$. Визначити максимальний елемент і зафіксувати його номер (рисунок 6.3).

Позначення: \max – максимальний елемент масиву;

k – номер максимального елементу в масиві.

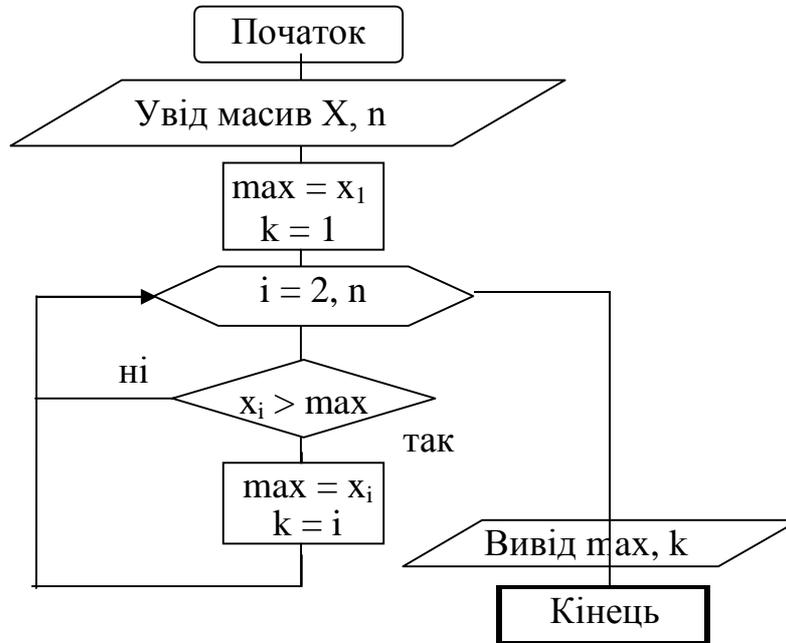


Рис. 6.3. Алгоритм пошуку максимального елементу та його номеру

Для визначення мінімального елементу масиву, слід внести зміни у блок-схему самостійно.

6.4. Обчислення середньоарифметичного та середньгеометричного елементів масиву

Приклад 6.4. Дан масив чисел $X = (x_1, x_2 \dots x_n)$. Обчислити

- а) середньоарифметичне значення додатних елементів;
- в) середньгеометричне значення елементів більших за **1**.

На рисунку 6.4. зображені блок-схеми алгоритмів

Позначення: S – сума елементів масиву

P – добуток елементів масиву

K – кількість елементів масиву

SA – середньоарифметичне елементів масиву

SG – середньгеометричне елементів масиву

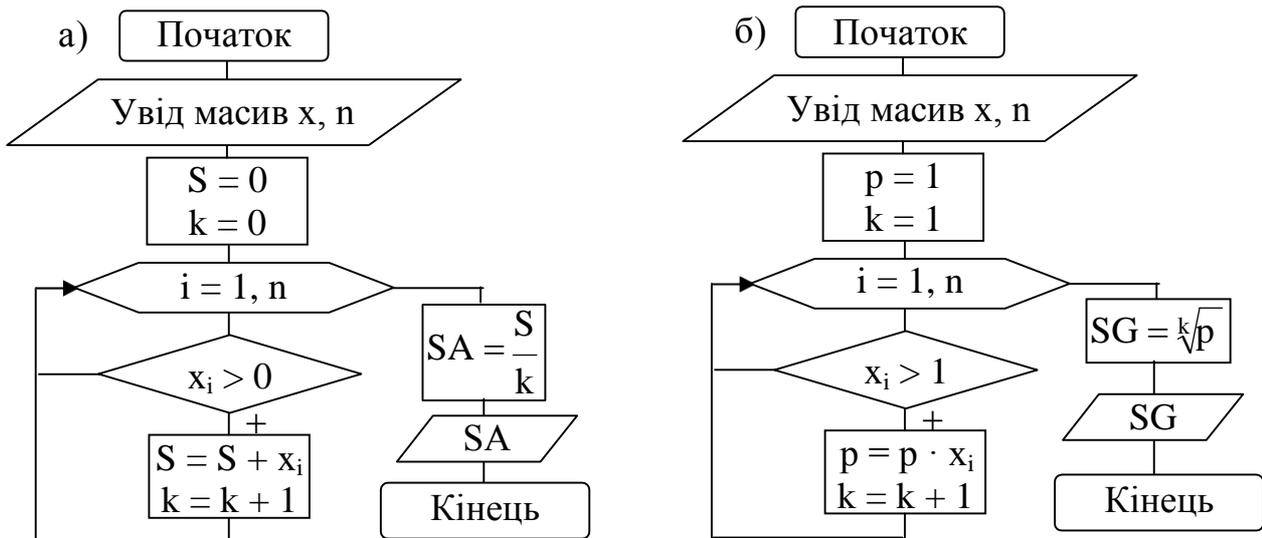


Рис. 6.4. а) Обчислення середньоарифметичного додатних елементів масиву;
 б) Обчислення середньгеометричного елементів більших за 1

6.5. Парні елементи, парні індекси елементів масиву

Приклад 6.5.1. Дан масив $X = (x_1, x_2 \dots x_n)$. Збільшити усі парні елементи масива у 2 рази.

Скористуємось визначенням цілої частки числа. Ціла частка числа – це найбільше ціле, яке не більше даного числа.

Наприклад: $[3,5] = 3$, $[3,9] = 3$, $[-2,2] = -3$.

$[]$ – визначення цілої частки числа.

На рисунку 6.5.1. зображена блок-схема алгоритму

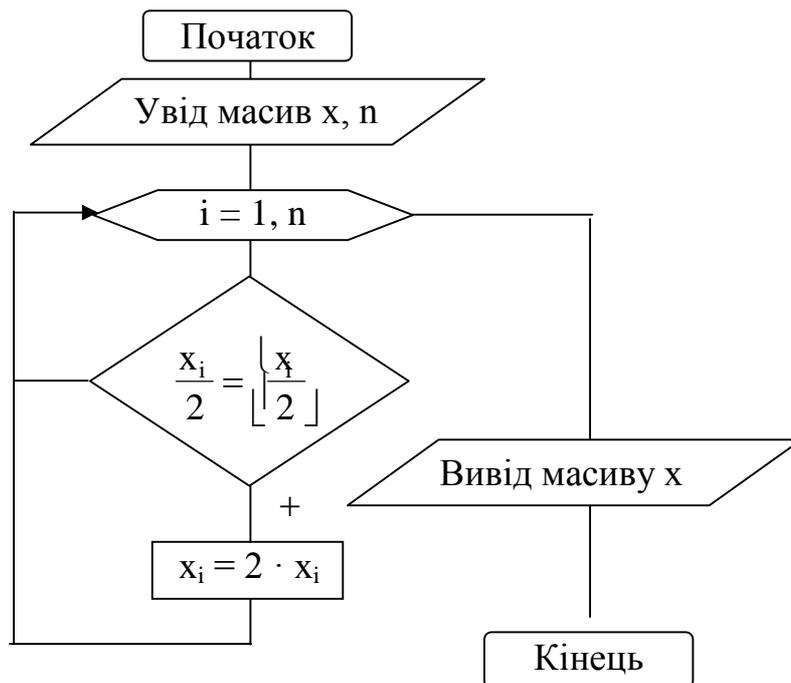


Рис. 6.5.1. Блок-схема алгоритму прикладу 6.5.1.

Приклад 6.5.2. Дан масив $X = (x_1, x_2, \dots, x_n)$. Елементи з парними номерами замінити на (-1) , а останні поділити на 1-ий елемент масиву.

На рисунку 6.5.2. зображена блок-схема алгоритма.

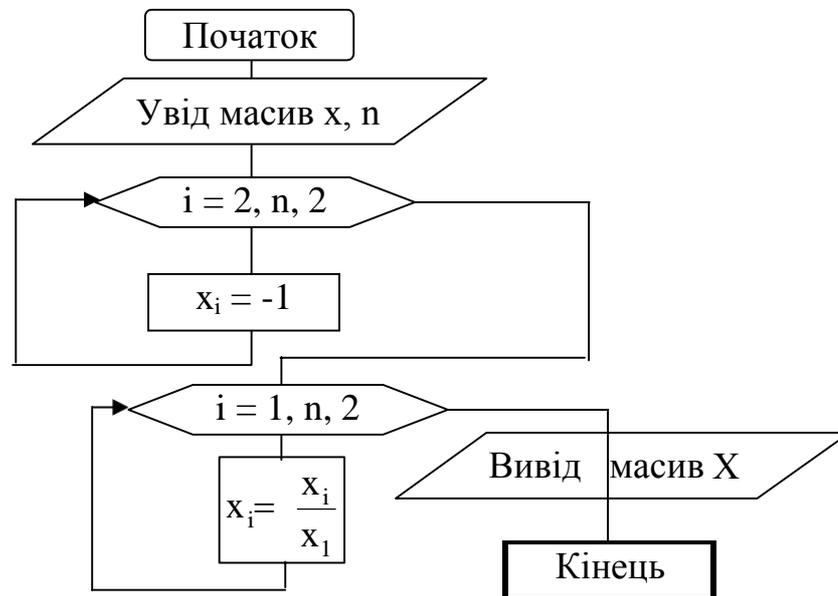


Рис. 6.5.2. Блок-схема алгоритма прикладу 6.5.2.

6.6. Сортування елементів масиву

Приклад 6.6. Дан масив $A = (a_1, a_2, \dots, a_n)$.

Отсортувати значення його елементів по зростанню.

Будемо зрівнювати два поруч розташованих елемента, якщо наступний елемент менший попереднього, то переставимо їх місцями.

Пояснення:

$r = 0$ – признак того, що перестановок не було;

$r = 1$ – признак того, що була хоча б одна перестановка.

На рисунку 6.6. зображена блок-схема алгоритму.

Алгоритм сортування елементів масиву по зменшенню їх значень скласти самостійно.

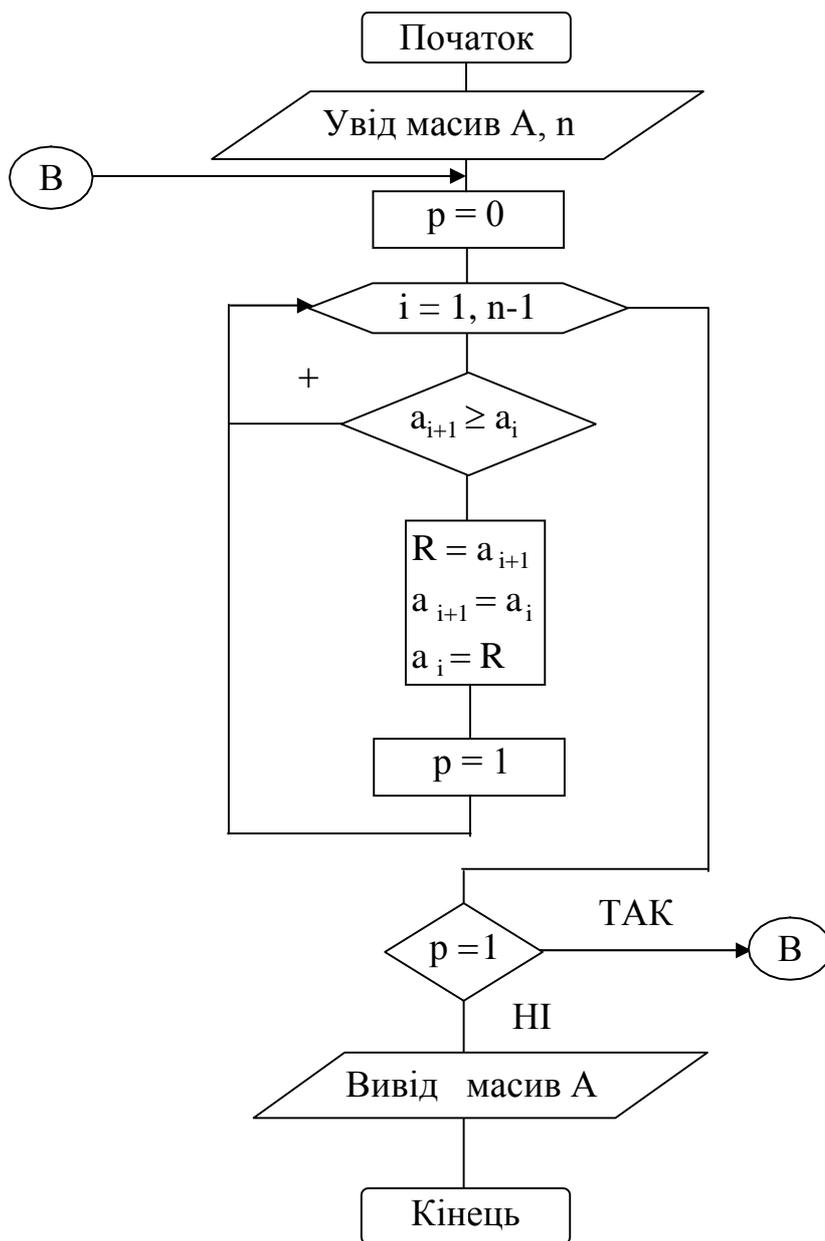


Рис. 6.6. Блок-схема алгоритму сортування елементів масиву по зростанню

6.7. Перезапис елементів одного масиву в іншій за заданою умовою

Приклад 6.7.1. Дан масив $A = (a_1, a_2, \dots, a_n)$.

Переписати у масив B ті елементи масиву A , які мають парні індекси.

Пояснення:

i – номер елементу в масиві B ;

j – номер елементу в масиві A ;

k – кількість елементів в масиві B .

На рисунку 6.7.1. зображена блок-схема алгоритма.

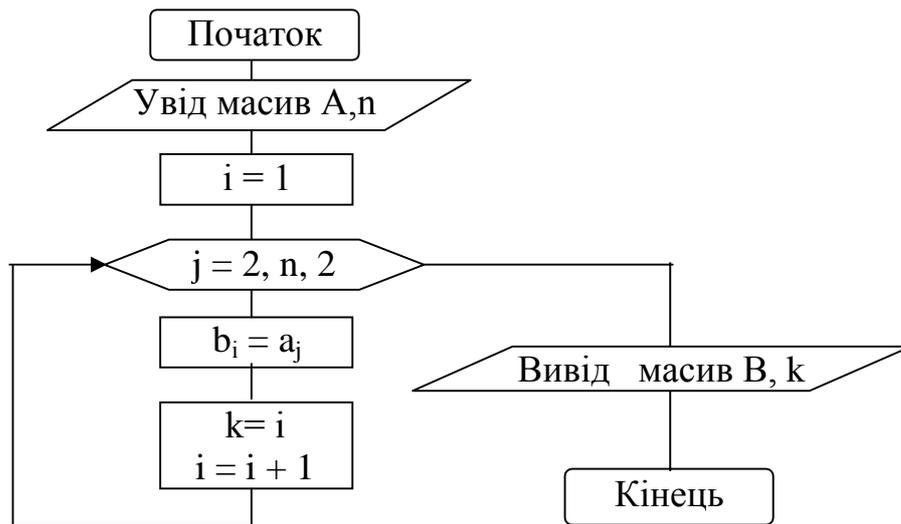


Рис. 6.7.1. Блок-схема алгоритму прикладу 6.7.1.

Приклад 6.7.2. Дан масив $A = (a_1, a_2, \dots, a_n)$. Переписати у масив B усі від'ємні елементи масиву A .

Пояснення:

- i – номер елементу в масиві B ;
- j – номер елементу в масиві A ;
- k – кількість елементів в масиві B .

На рисунку 6.7.2. зображена блок-схема алгоритму.

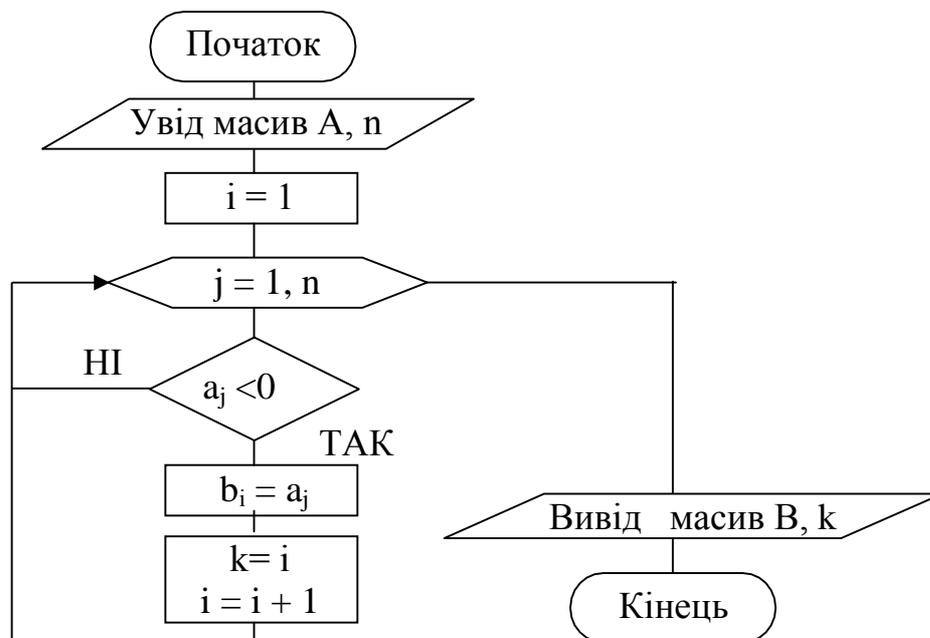


Рис. 6.7.2. Блок-схема алгоритму прикладу 6.7.2.

6.8. Приклади деяких алгоритмів

Приклад 6.8.1. Визначити найбільший з від'ємних елементів масиву $Y = (y_1, y_2, \dots, y_{30})$.

На рисунку 6.8.1. зображена блок-схема алгоритму.

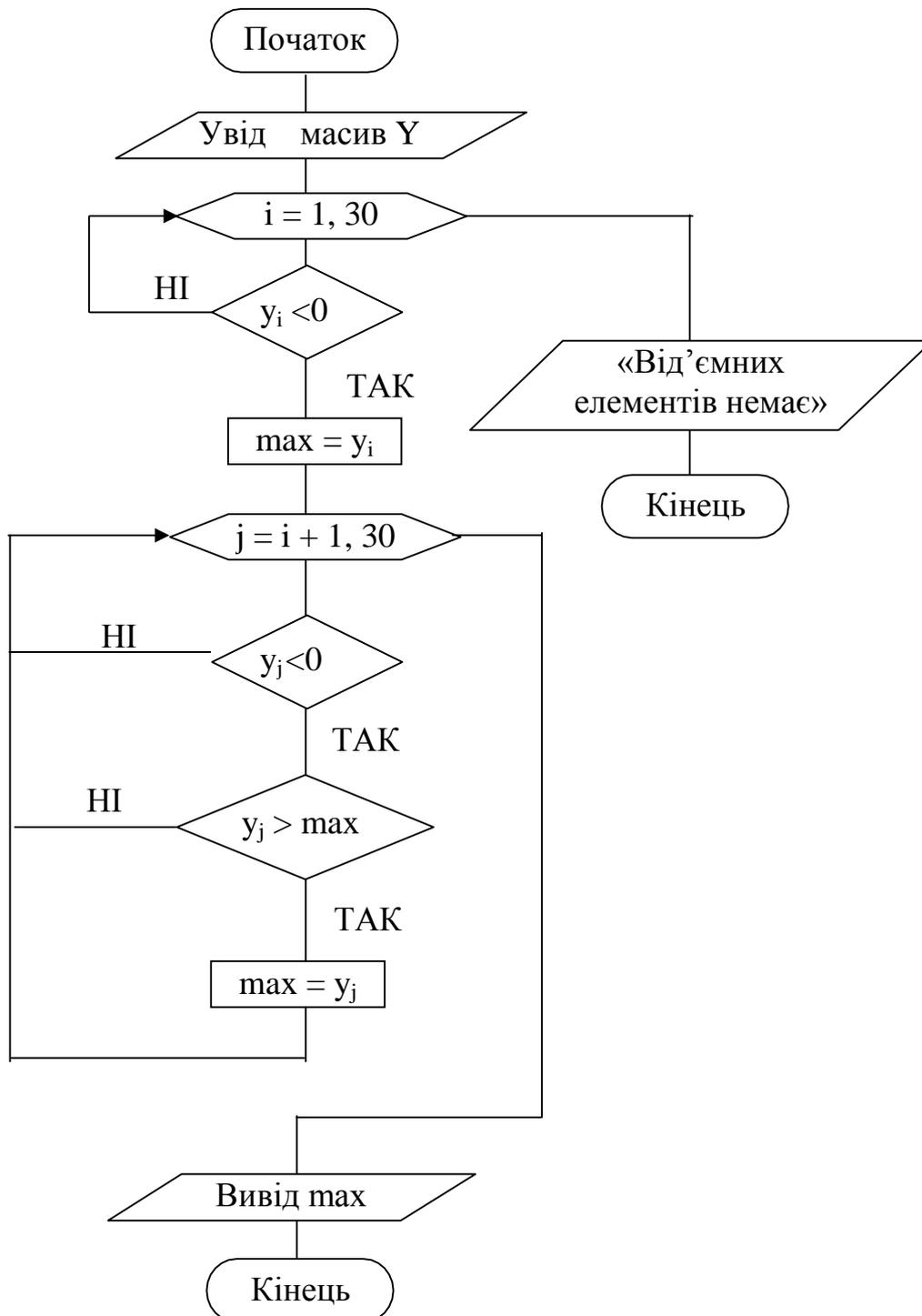


Рис. 6.8.1. Блок-схема алгоритму прикладу 6.8.1.

Приклад 6.8.2. Послідовність елементів a_1, a_2, \dots, a_n задана формулою

$$a_i = 2^i + 3^{i+1} + i, \quad i = \overline{1, n}, \quad n - \text{задане число.}$$

Визначити кількість елементів кратних 3.

На рисунку 6.8.2. зображена блок-схема алгоритму.

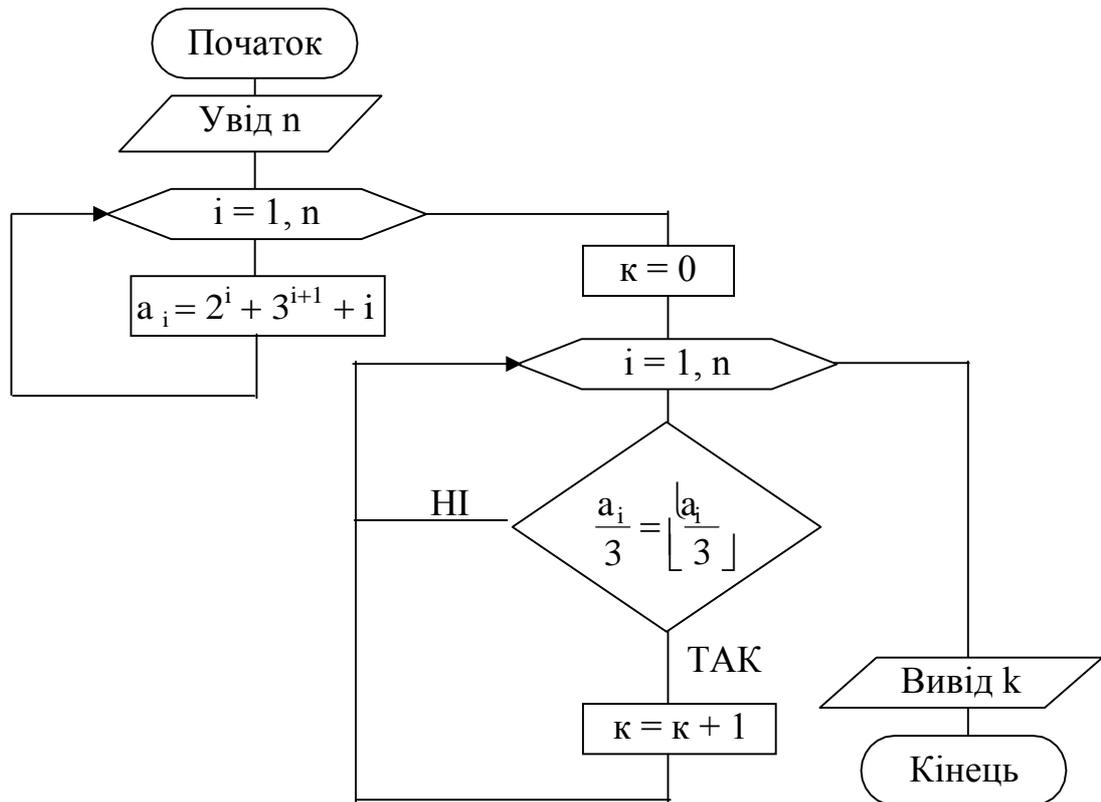


Рис. 6.8.2. Блок-схема алгоритму 6.8.2.

7. ОБЧИСЛЕННЯ СУМИ (ДОБУТКУ) РЯДУ

Приклад 7.1. Знайти суму ряду

$$\frac{y}{2} + \frac{y^2}{4} + \frac{y^3}{6} + \dots + \frac{y^{10}}{20}, \quad \text{де } y - \text{задане число.}$$

На рисунку 7.1. зображена блок-схема алгоритму.

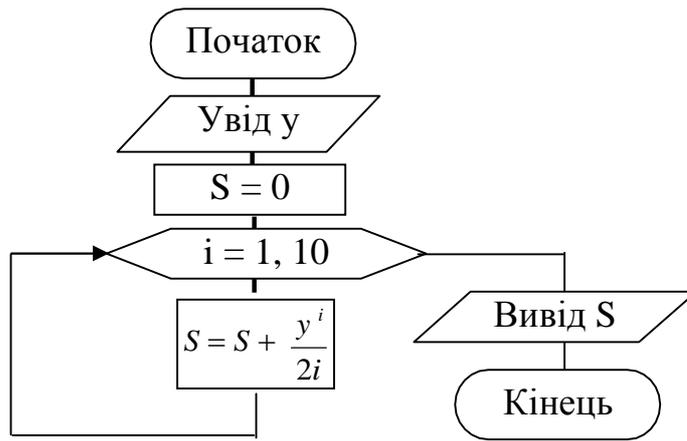


Рис. 7.1. Блок-схема алгоритму прикладу 7.1.

Приклад 7.2. Обчислити суму ряду

$$\frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} + \dots + \frac{1}{999 \cdot 1000}$$

На рисунку 7.2. зображена блок-схема алгоритму.

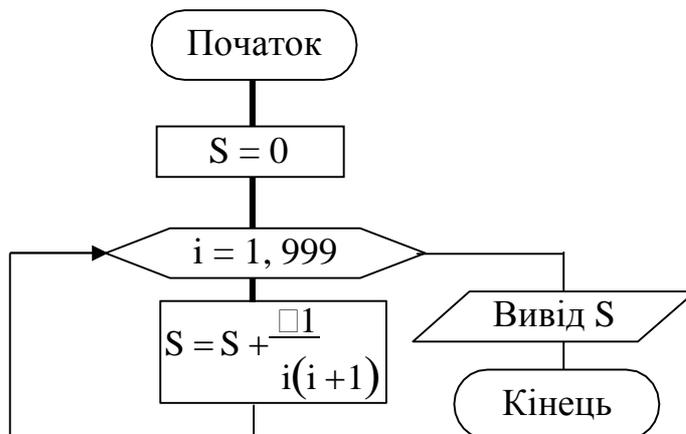


Рис. 7.2. Блок-схема алгоритму прикладу 7.2.

Приклад 7.3. Обчислити добуток за формулою

$$a(a - n)(a - 2n) \dots (a - n^2),$$

якщо a, n – задані числа.

На рисунку 7.3. зображена блок-схема алгоритму.

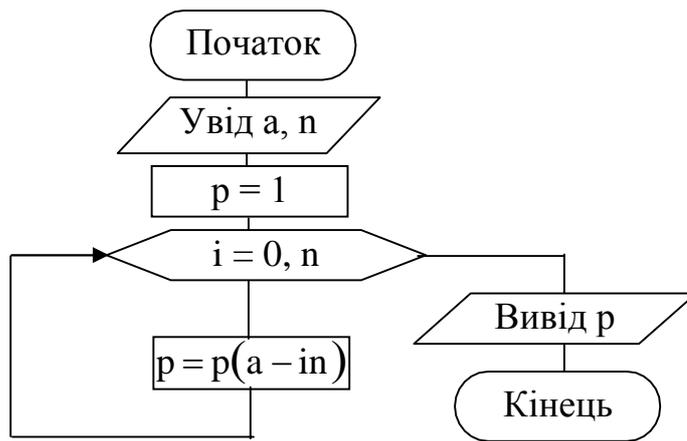


Рисунок 7.3. Блок-схема алгоритму прикладу 7.3.

8. ДВОМІРНІ МАСИВИ

Двомірний масив має вигляд:

$$A(m, n) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \text{ де}$$

m – кількість рядків
 n – кількість стовпців
 a_{ij} – елемент масиву
 i – номер рядка
 j – номер стовпця

Приклад 8.1. Обчислити суму елементів у кожному рядку заданого масиву $A(m, n)$.

m, n – задані значення.

На рисунку 8.1. зображена блок-схема алгоритму.

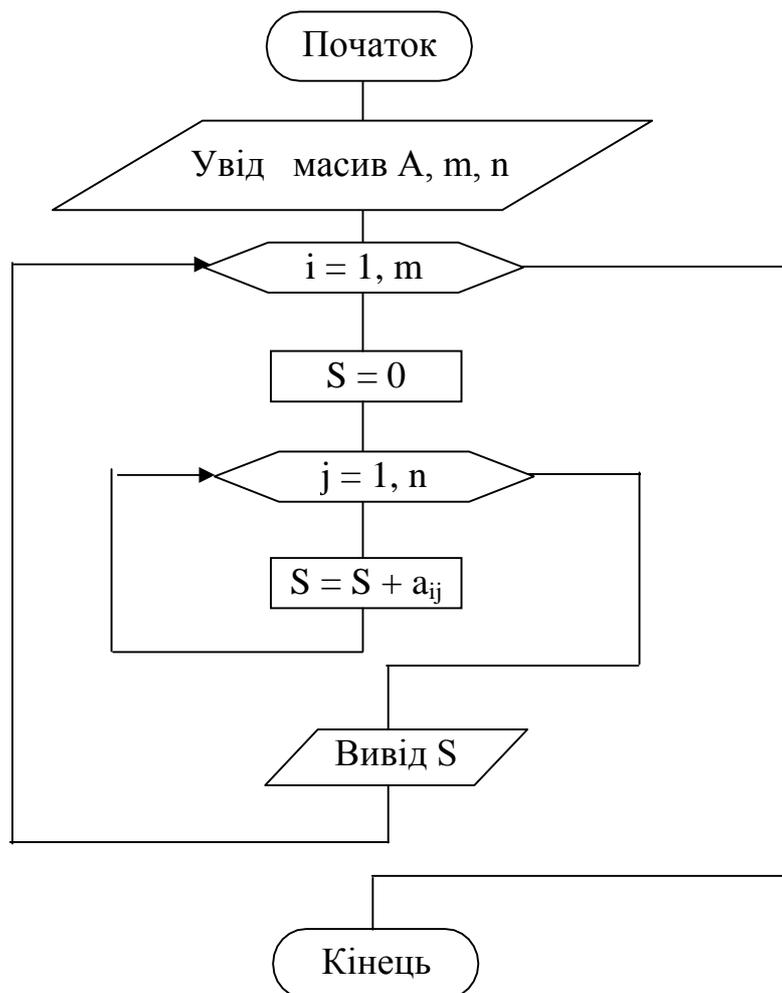


Рис. 8.1. Блок-схема прикладу 8.1.

Приклад 8.2. Визначити максимальний (max) елемент у кожному стовпцю двомірного масиву $A(m,n)$.

На рисунку 8.2. зображена блок-схема алгоритму.

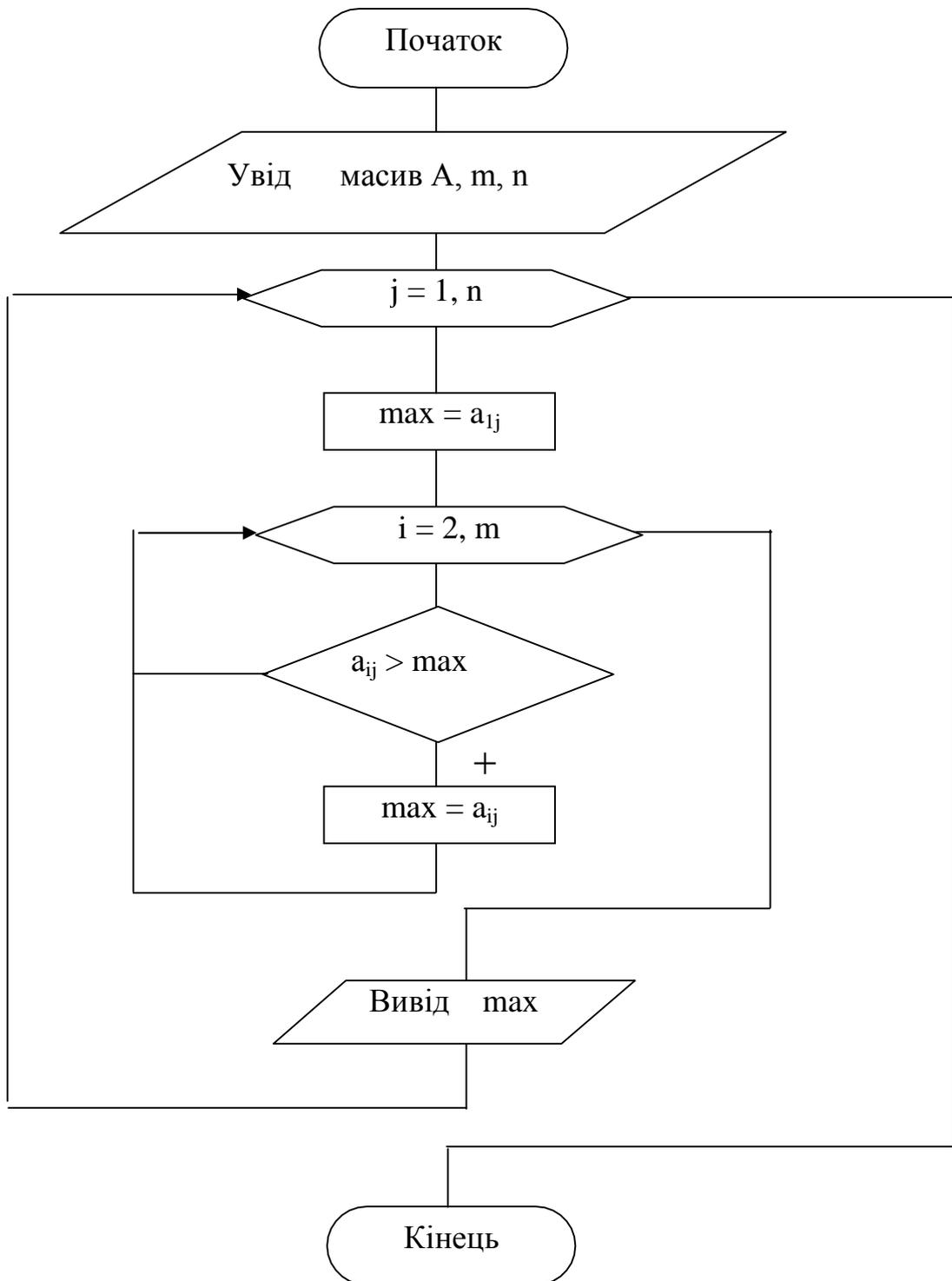


Рис. 8.2. Блок-схема алгоритму прикладу 8.2.

Приклад 8.3. Визначити суму елементів головної діагоналі і суму елементів додаткової діагоналі у квадратній матриці $B_{n,n}$.

Позначимо: S_1 – сума елементів головної діагоналі матриці
 S_2 – сума елементів додаткової діагоналі матриці

На рисунку 8.3. зображена блок-схема алгоритму.

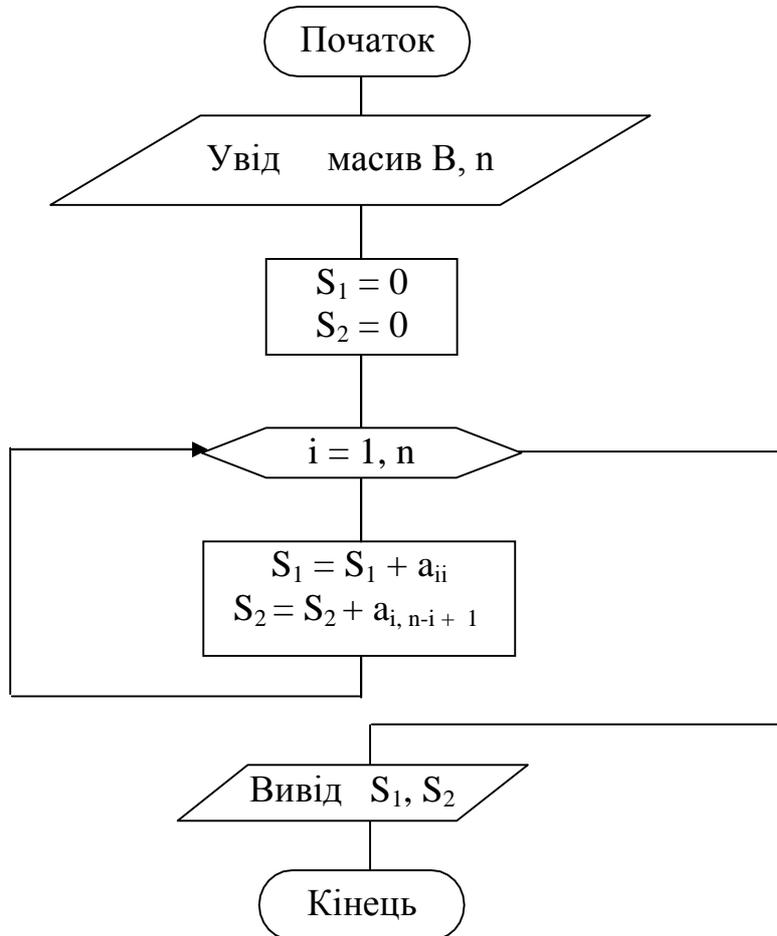


Рис. 8.3. Блок-схема алгоритму прикладу 8.3.

9. ПІДПРОГРАМИ

При створенні додатків часто застосовується структурне програмування. При цьому програма розбивається на підпрограми (підалгоритми), кожна з яких виконує одну з дій, які передбачені умовами задачі.

Важлива характеристика підпрограм — це можливість їх повторного виконання.

Підпрограми бувають 2-х видів — процедури та функції.

9.1. Підпрограми процедури

Приклад 9.1. Існують масиви $X = (x_1, x_2, \dots, x_{10})$ та $Y = (y_1, y_2, \dots, y_{20})$. Знайти суми додатних елементів у кожному з них.

На рисунку 9.1. зображені блок-схеми алгоритму.

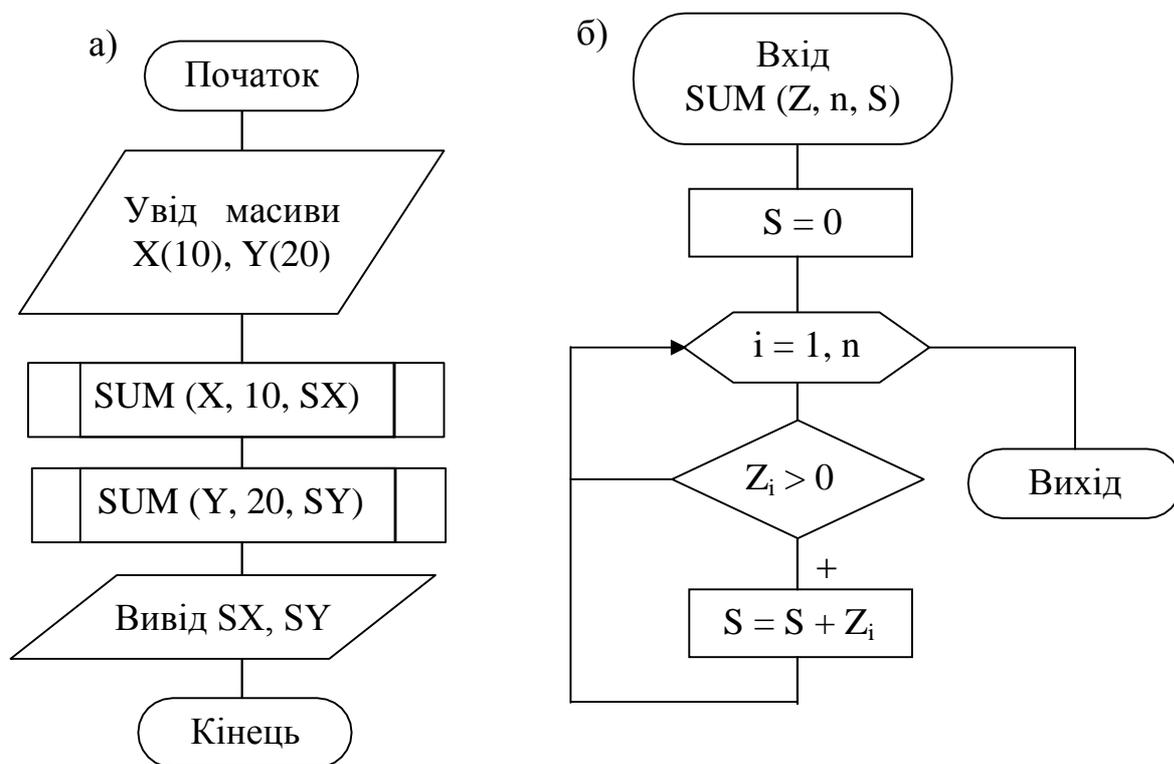


Рис. 9.1. Блок-схеми алгоритму прикладу 9.1.

а) блок-схема до головної програми

б) блок-схема до підпрограми процедури

SUM – ім'я підпрограми процедури

X, 10, SX – фактичні параметри

SX – сума додатних елементів масиву X

Y, 20, SY – фактичні параметри

SY – сума додатних елементів масиву Y

Z, n, S – формальні параметри

Z – одномірний масив

n – кількість елементів у масиві Z

S – сума додатних елементів у масиві Z

Приклад 9.2. Знайти кількість від'ємних елементів у кожному стовбці заданого двомірного масиву $A(3,4)$.

На рисунку 9.2. зображені блок-схеми алгоритму.

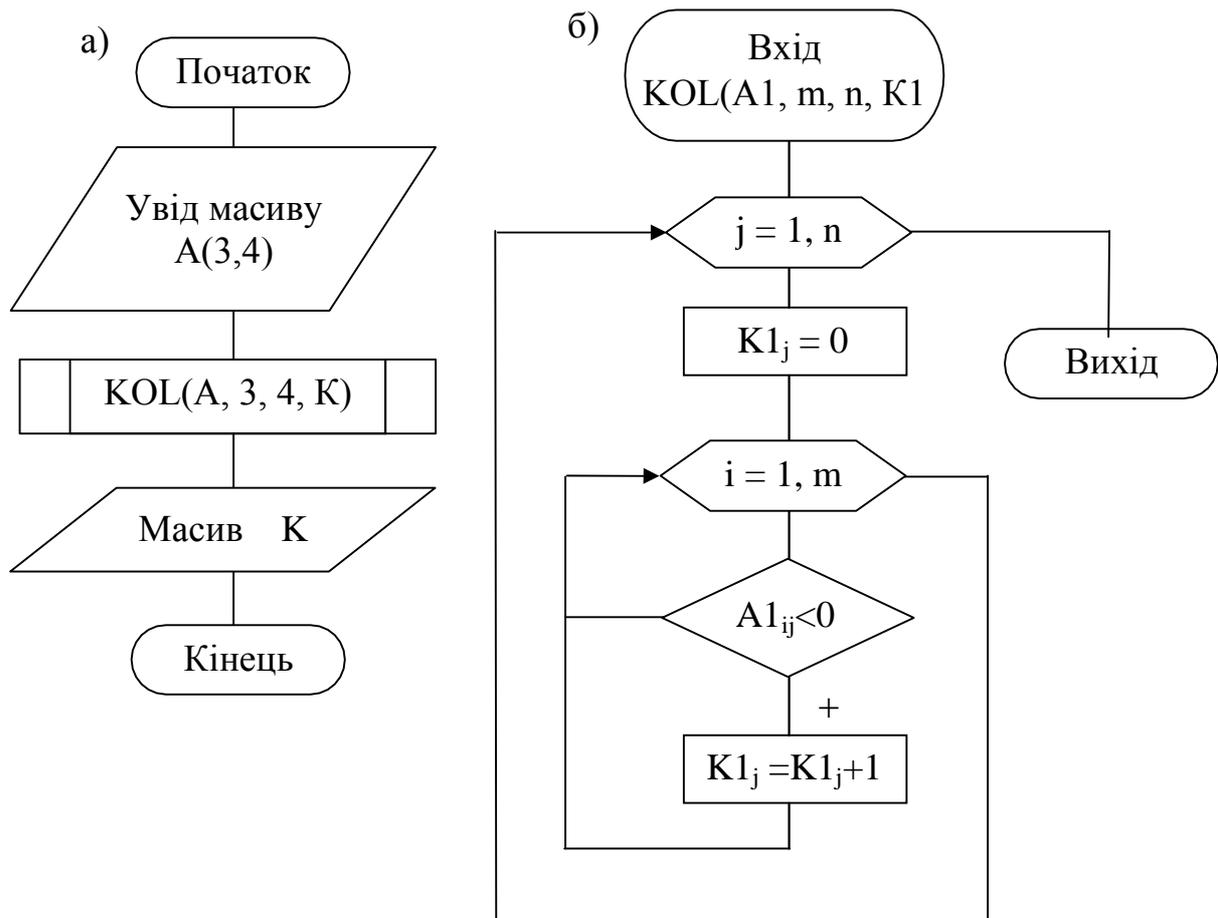


Рис. 9.2. Блок-схеми алгоритму прикладу 9.2.

а) блок-схема до головної програми

б) блок-схема до підпрограми процедури

KOL – ім'я підпрограми процедури

$A, 3, 4, K$ – фактичні параметри

K – ім'я одномірного масиву, який утримує кількість від'ємних елементів в кожному стовбці заданого масиву $A(3,4)$.

$A1, m, n, K1$ – формальні параметри

$A1$ – двомірний масив

m – кількість рядків

n – кількість стовбців

$K1$ – одномірний масив

9.2. Підпрограми функції

Приклад 9.3. Існують масиви $X = (x_1, x_2, \dots, x_{10})$ та $Y = (y_1, y_2, \dots, y_{20})$. Знайти максимальний елемент в кожному з них.

На рисунку 9.3. зображені блок-схеми алгоритму.

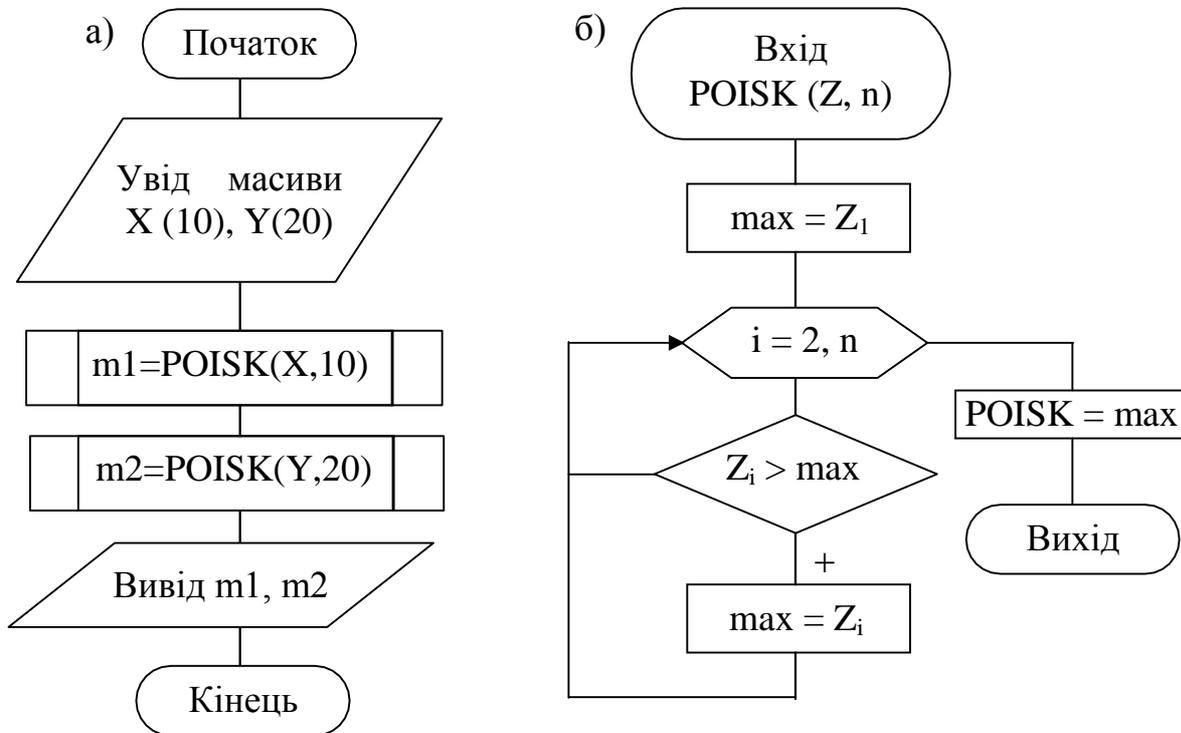


Рис. 9.3. Блок-схеми алгоритму прикладу 9.3.

а) блок-схема до головної програми

б) блок-схема до підпрограми функції

POISK – ім'я функції

$X, 10$ – фактичні параметри

$m1$ – максимальний елемент у масиві X

$Y, 20$ – фактичні параметри

$m2$ – максимальний елемент у масиві Y

Z, n – формальні параметри

Z – одномірний масив

n – кількість елементів у масиві Z

Приклад 9.4. Створити підпрограму функцію для обчислення значення функції:

$$Z = \begin{cases} 1 + 3,8 \cos x, & x \geq 2 \\ \sqrt{x^2 + 2y^2}, & x < 2 \end{cases}$$

$$x = b \sin \alpha; \quad y = 8,6 + x^2; \quad b = 13,4; \quad \alpha \in [0,1;55]; \quad \Delta \alpha = 1,2.$$

На рисунку 9.4. зображені блок-схеми алгоритму.

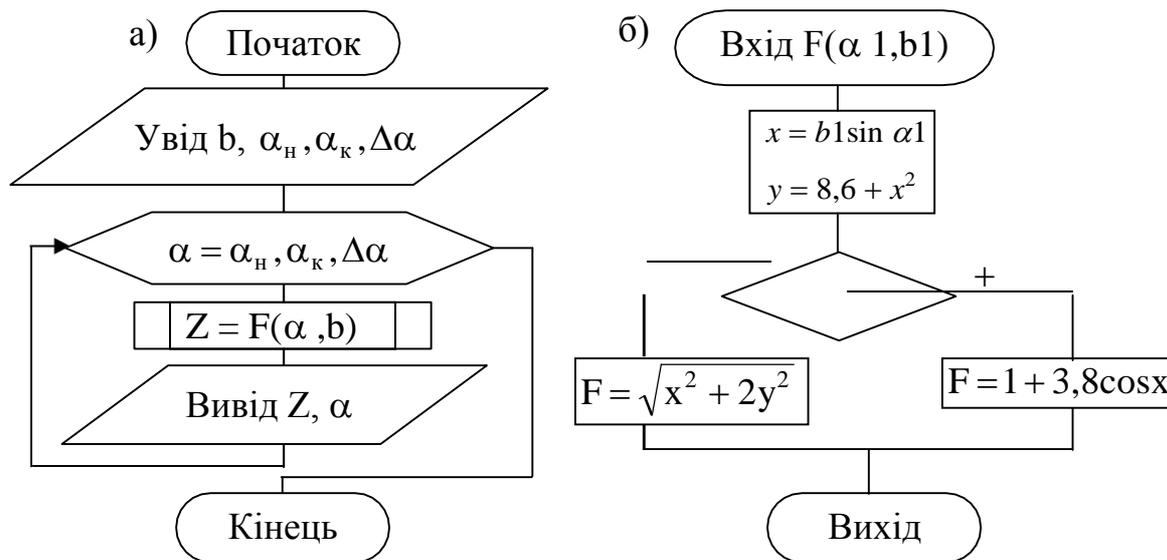


Рис. 9.4. Блок-схеми алгоритму прикладу 9.4.

а) блок-схема до головної програми

б) блок-схема до підпрограми функції

F – ім'я функції

α , b – фактичні параметри

$\alpha 1$, b1 – формальні параметри

10. ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОЇ РОБОТИ

10.1. Алгоритми лінійної структури

Скласти блок-схеми алгоритмів обчислення значення функції Y за варіантами:

$$1. y = \frac{x^4 - bx^3 - a}{(x+a)(x-b)}, \quad \partial e \ b = 2^x; \ x = \lg 0,05 + 2; \ a = b + 2$$

$$2. y = \frac{x + 3a - k_1}{k_1 x + k_2}, \quad \partial e \ x = 7a + k_1 \cdot k_2; \ k_1 = 0,8; \ k_2 = 4,8; \ a = 0,25$$

$$3. y = \frac{\sin^3 ax + b}{\cos^2 x}, \quad \partial e \ x = -3,8; \ a = 0,5c + x^2; \ c = \ln 0,08; \ b = x^2 + c$$

$$4. y = \sin \frac{a}{x} + \lg 0,08e^x, \quad \partial e \ x = -2,5a^2; \ a = 0,8b + c; \ c = 0,5; \ b = c^3$$

$$5. y = \frac{(a\sqrt[3]{b} - cd)^2}{a + b + c}, \quad \partial e \ a = 7,14; \ b = a^2 - 1; \ c = a^3 - b^3; \ d = \sqrt{(b-c)^3}$$

$$6. y = \frac{\alpha_1 J_1 - \alpha_2 J_2}{\alpha_1 + J_1^2}, \quad \partial e \ \alpha_1 = \sin 0,18; \ J_1 = e^4; \ \alpha_2 = \ln 0,5 + J_1; \ J_2 = 1,7 \cdot J_1$$

$$7. y = \frac{(a+b)^2 + m}{1 + x^a + b^a}, \quad \partial e \ x = 0,81; \ a = \lg 0,83; \ b = e^a; \ m = x - 1$$

$$8. y = (1+z) \frac{x+y}{a-x}, \quad \partial e \ x = 0,8 \cdot 10^{-2}; \ y = e^{\sqrt{x^3}}; \ z = 5 \cdot 10^{-3}; \ a = z + \frac{y}{2}$$

$$9. y = \frac{\sqrt{x^4 + ax + b}}{\sqrt[3]{x^4 - ax - b}}, \quad \partial e \ x = 0,75; \ a = -x^2 + \lg 0,08; \ b = e^{-x} + a$$

$$10. \quad y = \ln(e^x + bx^2), \quad \partial e x = \sin^2(a + b); \quad a = 0,36; \quad b = \sin^2 a$$

$$11. \quad y = 5\sin^2 |\ln(cx + 1)|, \quad \partial e c = 4,8; \quad a = 3,6; \quad x = \cos^2(a + b); \quad b = \sin a$$

$$12. \quad y = \sqrt{x^3 + ax^2} + c, \quad \partial e x = 0,36; \quad a = x + 0,52b; \quad b = ax^2 + 1; \quad c = 0,7$$

$$13. \quad y = \cos \frac{1}{x + 0,2} + \lg 0,8x, \quad \partial e x = -2,6a^2 + b^3\sqrt{c}; \quad c = 7,14; \quad a = 0,8c; \quad b = ac^2$$

$$14. \quad y = \frac{(b^4\sqrt{c} - bd)^2}{a - b}, \quad \partial e c = 0,13; \quad b = c^3 - 4a; \quad a = 7,6; \quad d = c^2 + b^2$$

$$15. \quad y = \sin^3 \left(\frac{x+a}{2} \right) - \cos x, \quad \partial e x = 5b + e^a; \quad a = 4,8; \quad b = \ln a$$

10.2. Алгоритми розгалуженої структури

Скласти блок-схеми алгоритмів обчислення значення функції Y за варіантами:

$$1. \quad y = \begin{cases} x^2 - \sin \gamma, & x \leq 0 \\ \sqrt{x} + \cos \gamma, & x > 0 \end{cases};$$

$\gamma = 0,35a; a = x + 3; x = 1,8.$

$$2. \quad y = \begin{cases} 4x^3 + \gamma, & x \leq 2 \\ (x + 3\gamma) \cdot x, & x > 2 \end{cases}$$

$x = 1,4; a = 3,5x; \gamma = \cos(a + x).$

$$3. \quad y = \begin{cases} x^3 + a, & x < 0 \\ \sin \frac{x}{a}, & x = 0 \\ \sqrt{x} + \frac{a}{2}, & x > 0 \end{cases}$$

$x = 1,3 - \ln a; a = 0,27.$

$$4. \quad y = \begin{cases} e^{-x+2} + a, & x \geq 1 \\ \sin(x+3,2), & x < 1 \\ a + 3 \end{cases},$$

$a = 14,8; x = e^a.$

$$5. \quad y = \begin{cases} 1, & x \geq 0 \\ \operatorname{tg} \frac{1}{x^2 + 8,2}, & x < 0 \\ \frac{0,32x}{x^2 + 3}, & x < 0 \end{cases}$$

$x = 0,5k + b; b = \sin 1,7; k = -0,8.$

$$6. \quad y = \begin{cases} 9bx, & x < 2 \\ x - 2bx^2, & x < 2 \\ \cos(b + x), & x \geq 2 \end{cases},$$

$x = \ln 0,7; b = 0,3.$

$$7. \quad y = \begin{cases} \cos x - \sin^3 x, & x \geq 1,5 \\ xe^{-x}, & x < 1,5 \end{cases}$$

$x = 0,9 \cdot \sin a; a = e^{0,17}.$

$$8. \quad y = \begin{cases} x(A - C), & A = C \\ x^3 - A, & A > C \\ x^3 + A, & A < C \end{cases}$$

$x = 8,6; a = 2\cos x; C = A + 1.$

$$9. \quad y = \begin{cases} 0,5\cos x, & x < 1 \\ 0,25x^a, & x = 1 \\ 0,9\sqrt{x}, & x > 1 \end{cases}$$

$x = 1,7 - e^{0,35}.$

$$10. \quad y = \begin{cases} 6z^2 - 5, & x \leq 1 \\ 5z^3 + 1, & x > 1 \end{cases}$$

$z = 0,5; x = \arccos z.$

$$11. \quad y = \begin{cases} (4-x)^2, & b < 5 \\ 0,25 + bx, & b \geq 5 \end{cases}$$

$x = 0,7; b = cx^2; c = \sin^2 x.$

$$12. \quad y = \begin{cases} a + b - x, & x \geq 1,5 \\ \sin^2 x + \cos(a + b), & x < 1,5 \end{cases}$$

$a = 1,3; b = 0,85; x = 4,7 - \lg a.$

$$13. \quad y = \begin{cases} 2,35 + x^3, & x \leq 1,5 \\ \sqrt{0,85x^2}, & x > 1,5 \end{cases}$$

$x = 5a + \sin 3,8; a = 14,6.$

$$14. \quad y = \begin{cases} \frac{a}{2}(1-\alpha) + x^3, & \alpha \leq 1 \\ 0,1\alpha^4, & \alpha > 1 \end{cases}$$

$\alpha = 10,2; a = 0,35\alpha.$

$$15. \quad y = \begin{cases} \sin^2 \lambda - x^2, & x \leq 0 \\ \cos \lambda + \sqrt{x^3 + 1}, & x < 0 \end{cases}$$

$\lambda = 0,35; x = \lambda^2 + a^2; a = 10.$

3. Алгоритми циклічної структури

Скласти блок-схеми алгоритмів обчислення значення функції Y за варіантами:

1.	$Z = \begin{cases} 1 + 2,5e^{-0,8x}, & x > 1 \\ \sqrt{x^2 + 2y^2}, & x \leq 1 \end{cases}$ $x = a \sin \alpha; y = 5,6 - 2x; a = 8,3; \alpha_{\Pi} = 0,25; \alpha_{\kappa} = 0,50; \Delta\alpha = 0,05.$
2.	$Z = \begin{cases} 8,6 \ln 3,5(1 + x^2), & x < 1,5 \\ \sqrt[3]{4,8 + y^2}, & x \geq 1,5 \end{cases}$ $x = e^{1,5\alpha}; y = ax^2 - 3,6; a = 2,8, \alpha_{\Pi} = 0,35; \alpha_{\kappa} = 0,75; \Delta\alpha = 0,05.$
3.	$Z = \begin{cases} 7,2x^2 \operatorname{tg} \alpha, & y > 1 \\ (1 + 2y^2) \cos \alpha, & y \leq 1 \end{cases}$ $x = a \cdot e^{3\alpha}; y = 2x \cdot \cos \alpha; a = 3,7, \alpha_{\Pi} = 0,30; \alpha_{\kappa} = 0,60; \Delta\alpha = 0,05.$
4.	$Z = \begin{cases} 7,5x^2 + 2,8y^2, & x < 3 \\ 4,5 - \sqrt{x^2 + 3,4y^2}, & x \geq 3 \end{cases}$ $x = a \sin \alpha; y = x^2 \cos \alpha; a = 4,5; \alpha_{\Pi} = 0,20; \alpha_{\kappa} = 0,70; \Delta\alpha = 0,10.$
5.	$Z = \begin{cases} \sqrt{6,7x^2 + 2y^2}, & x < 1 \\ 3,8e^{-0,6x}, & x \geq 1 \end{cases}$ $x = 1 + 2,7 \operatorname{tg} \alpha; y = a(1 + 3,5 \cos 2\alpha); a = 0,30; \alpha_{\Pi} = 0,30; \alpha_{\kappa} = 0,60; \Delta\alpha = 0,05.$
6.	$Z = \begin{cases} \sqrt{2,5y + x^2}, & x < 2 \\ 6,7 \ln y , & x \geq 2 \end{cases}$ $x = t \operatorname{tg} \alpha; y = \sqrt{\frac{-x^2}{\cos \alpha}}; t = 6,75; \alpha_{\Pi} = 0,25; \alpha_{\kappa} = 0,48; \Delta\alpha = 0,03.$
7.	$Z = \begin{cases} x^2 \sin \alpha + y^2, & y > 1 \\ \sqrt{x^2 + y^2} \operatorname{tg} \alpha, & y \leq 1 \end{cases}$ $x = a^2 e^{1,5\alpha}; y = 3,8 \ln 5\alpha; a = 3,4; \alpha_{\Pi} = 0,30; \alpha_{\kappa} = 0,60; \Delta\alpha = 0,05.$
8.	$Z = \begin{cases} 3,5 + \sqrt[3]{16,8 + 8,4 \cdot x^2}, & x < 2 \\ (3,5 + xy)e^x, & x \geq 2 \end{cases}$ $x = (1 + \operatorname{tg}^2 \alpha) \cdot a; y = 2,5x^2 - 6,7; \alpha_{\Pi} = 0,30; \alpha_{\kappa} = 0,80; \Delta\alpha = 0,05; a = 2,3.$

9.	$Z = \begin{cases} 7,8e^x \cdot \sin \alpha, & y < 1 \\ \sqrt{1,6 + x^2} \cdot y, & y \geq 1 \end{cases}$ $x = (3,7 - a^2) \cos \alpha; y = a \sin \alpha; \alpha_{\Pi} = 0,40; \alpha_{\kappa} = 0,90; \Delta \alpha = 0,05; a = 7,6.$
10.	$Z = \begin{cases} (x^2 + y^2), & x < 1,5 \\ \frac{xy}{(x^2 - y^2)e^{xy}}, & x \geq 1,5 \end{cases}$ $x = 8,6 \sin \alpha \quad y = 3,5 t x \cos \alpha; \alpha_n = 0,35; \alpha_k = 0,75; \Delta \alpha = 0,05; t = 1,9.$
11.	$Z = \begin{cases} \sqrt{x^2 + y^2} e^{2x}, & x > 1 \\ 7xy, & x \leq 1 \end{cases}$ $x = a * \operatorname{tg}(\alpha)^2; y = (1 + x) * \cos(2\alpha); \alpha_n = 0,25; \alpha_k = 0,50; \Delta \alpha = 0,05; a = 9,2.$
12.	$Z = \begin{cases} \sqrt{1 + xy}, & x < 1 \\ (x^2 + y^2)e^{xy}, & x \geq 1 \end{cases}$ $x = a \sin \alpha; y = 3,8a + \sqrt{atg\alpha}; \alpha_{\Pi} = 0,35; \alpha_{\kappa} = 0,75; \Delta \alpha = 0,05; a = 7,3.$
13.	$Z = \begin{cases} x \sin \alpha + y \cos \alpha, & y > 2 \\ ye^x, & y \leq 2 \end{cases}$ $x = \sqrt{2,5 + a^2 \operatorname{tg} \alpha}; y = a \ln \alpha; \alpha_{\Pi} = 0,10; \alpha_{\kappa} = 0,60; \Delta \alpha = 0,1; a = 5,8.$
14.	$Z = \begin{cases} x^2 \sin \alpha + y^2 \cos \alpha, & x > 2 \\ \sqrt{xy + 3,6 \operatorname{tg} \alpha}, & x \leq 2 \end{cases}$ $x = e^{2,5a}; y = 1 + a^2 x^2; \alpha_{\Pi} = 0,14; \alpha_{\kappa} = 0,32; \Delta \alpha = 0,04; a = 18,6.$
15.	$Z = \begin{cases} (x^2 - y^2)e^x, & y > 1 \\ 4,6 + x^2 y^2 \sin \alpha, & y \leq 1 \end{cases}$ $x = 7,6 \ln a; y = \operatorname{tg}^2 \alpha; \alpha_{\Pi} = 0,35; \alpha_{\kappa} = 0,6; \Delta \alpha = 0,05; a = 1,4.$

10.4. Одномірні масиви

Скласти блок-схеми алгоритмів за варіантами:

Номер варіанта	Знайти:	Масив
1	Середнє арифметичне S додатних елементів і їхню кількість K .	-45,9; 2,82; 30,317; -43,5; 34,2; 13,2; 0; -10,4; 30,874
2	Суму $S1$ елементів із парними номерами і суму $S2$ елементів із непарними номерами.	17,76; -12,7 ; -16,8; 14,16; 36,325; 9,547; 15,796; -19,301; 0; 5,3; 6,8; -17,3
3	Суму S квадратів усіх елементів, що перевищують 10 по абсолютному значенню і їхню кількість K .	42,027; 0; -23,018; 0; -18,532; 0,73; 30,8; 39,115; 7,3; -18,67; 12,32; -8,05; -16,3
4	Добуток $P1$ усіх додатних елементів і їхню кількість $K1$, а також, добуток $P2$ усіх від'ємних елементів і їхню кількість $K2$.	43,175; -11,082; 0; 32,217; -5,42; -2,477; 13,921; -14,184; -7,3; 8,13; 16,08; 123,3; 18,67
5	Середнє арифметичне S квадратів усіх елементів, що перевищують 2,5 по абсолютному значенню та їхню кількість K .	-10,396; -3,47; -14,748; 0; -2,34; 43,796; -2,616; 46,139; 0,35; 5,75; -1,308; 7,87
6	Добуток відмінних від нуля елементів із непарними номерами і їхню кількість $K1$.	16,375; -17,004; -49,399; -43,353; -15,530; -3,001; 21,762; -42,420; 7,375; -0,675; 13,834; -7,68
7	Суму S квадратів елементів, значення яких належать відрізку $(-10,12)$ і їхню кількість K .	-13,27; -9,547; 0; -22,477; 43,796; -3,001; -28,706; 9,488; -41,66; 13,879; 16,713

8	Середнє арифметичне S елементів, що не перевищують 15 по абсолютному значенню і їхню кількість K .	-28,221; 2,829; -18,7; -12,784; 0; -34,719; -17,04; -12,784; 1,89; 5,83; 56,13; -14,8
9	Значення M найбільшого елемента і його номер.	49,624; -20,481; 87,68; 0; 32,646; 118,37; 18,34; 5,68
10	Кількість N від'ємних, кількість P додатних і кількість Z нульових елементів.	11,749; 0; 0; -39,144; 0; 9,488; 21,412; 41,643; 5,6; 7,8; -0,34; -1,2; 17,13
11	Середнє арифметичне елементів, значення яких належать інтервалу $(-273; 20)$, і їхню кількість.	206,8; -31,18; 0; 36,9; -313,8; 0,67; -230,2; 0; 21,18; 5,64; 115,36; -270,3; 18,8; 35,7; 6,4
12	Значення M найбільшого по абсолютному значенню елемента і його номер N .	-41,5; -22,174; 40,464; 42,347; -10,089; -41,66; 40,843; 0; 20; -47,591; 34,458; 7,83
13	Суму S квадратів від'ємних елементів з номерами, кратними трьом, і кількість додатних елементів.	35,066,0,276; -13,94; 13,879; 8,73, 0; -13,762; -29,777; 45,194; -25,613; 38,642
14	Середнє арифметичне S елементів, відмінних від нуля і їхню кількість K_1 , а також кількість K_0 елементів, рівних нулю.	-34,22; 36,325; -18,532; -5,42; 0; -23,401; -15,53; 0; 0; -0,089; -13,94; 0; 0; -13,914
15	Добуток P відмінних від нуля елементів і кількість елементів рівних нулю.	-10,423; -19,301; 39,115; 0; 4,184; 0; 0; 46,139; 42,42; 0; 20; 0; 0

ЛАБОРАТОРНА РОБОТА №2

СИСТЕМИ ЧИСЛЕННЯ. ПРЕДСТАВЛЕННЯ РІЗНИХ ТИПІВ ЧИСЕЛ.

Мета роботи: Вивчити правила запису чисел у різних СЧ, правила переведення чисел, алгоритми кодування чисел.

Завдання до роботи:

1. Записати зображення чисел від 0 до 20 у десятковій, двійковій, вісімковій, шістнадцятковій системах числення.
2. Вивчити правила переведення цілих чисел з десяткової системи у двійкову і навпаки. Перевести у двійкову систему числення десяткові числа: 1,10,20,100,200,1000,32768. Перевести у десяткову систему числення двійкові числа.: 1,101,10000,1000101010,11001011.
3. Вивчити правила переведення дробових чисел з десяткової системи в двійкову і навпаки. Перевести у двійкову систему числення десяткові числа: 0,25; 0,625; 0,854; 3,768. Перевести у десяткову систему числення двійкові числа: 0,00011; 0,001; 0,10101010.
4. Вивчити правила переведення чисел із двійкової системи у вісімкову та шістнадцяткову і навпаки.
5. Вивчити алгоритм утворення прямого, оберненого та доповняльного кодів цілих чисел. Записати у прямому, оберненому та доповняльних кодах при 8-бітовому кодуванні десяткові числа та знайти їх суму в оберненому та доповняльному кодах: -55 і 203.

Зміст звіту:

1. Постановка завдань.
2. Правила та алгоритми виконання завдань.
3. Розв'язки індивідуальних завдань.

Контрольні запитання:

1. Які є види систем числення?
2. Як перевести десяткове число у двійкову систему?
3. Як перевести двійкове число у шістнадцяткову систему?
4. Від чого залежать межі діапазону цілих чисел при кодуванні у пам'яті ПК?
5. Для чого і як використовується доповняльний код?
6. Які можливі помилки при виконанні арифметичних дій у обмеженій кількості розрядів і як вони усуваються?
7. Як кодуються дійсні числа?

Варіанти індивідуальних завдань

1	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 987 2) $16 \rightarrow 2;10$: 7FE 3) $8 \rightarrow 2;10$: 754 4) $2 \rightarrow 8;10;16$: 10111011 5) $2 \rightarrow 10;8;16$: 0,10111 6) $10 \rightarrow 8$: 0,65</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах: 140 і -226</p> <p>3. Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>	9	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 154 2) $16 \rightarrow 2;10$: 8AC 3) $8 \rightarrow 2;10$: 704 4) $2 \rightarrow 8;10;16$: 10101100 5) $2 \rightarrow 10;8;16$: 0,11001 6) $10 \rightarrow 8$: 0,44</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах: -117 і 96.</p> <p>3. Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>
2	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 786 2) $16 \rightarrow 2;10$: F9A 3) $8 \rightarrow 2;10$: 373 4) $2 \rightarrow 8;10;16$: 11100011 5) $2 \rightarrow 10;8;16$: 0,01101 6) $10 \rightarrow 8$: 0,96</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах: -31 і 114</p> <p>3. Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>	10	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 811 2) $16 \rightarrow 2;10$: 89D 3) $8 \rightarrow 2;10$: 507 4) $2 \rightarrow 8;10;16$: 11001001 5) $2 \rightarrow 10;8;16$: 0,001101 6) $10 \rightarrow 8$: 0,59</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах: 14 і -126</p> <p>3. Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>
3	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 591 2) $16 \rightarrow 2;10$: 6EA 3) $8 \rightarrow 2;10$: 546 4) $2 \rightarrow 8;10;16$: 10111110 5) $2 \rightarrow 10;8;16$: 0,10011 6) $10 \rightarrow 8$: 0,84</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах: 200 і -98</p> <p>3. Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>	11	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 457 2) $16 \rightarrow 2;10$: E4B 3) $8 \rightarrow 2;10$: 107 4) $2 \rightarrow 8;10;16$: 11110011 5) $2 \rightarrow 10;8;16$: 0,11101 6) $10 \rightarrow 8$: 0,71</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах: -57 і 102</p> <p>3. Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>
4	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 308 2) $16 \rightarrow 2;10$: BE5 3) $8 \rightarrow 2;10$: 674 4) $2 \rightarrow 8;10;16$: 01110111 5) $2 \rightarrow 10;8;16$: 0,00111 6) $10 \rightarrow 8$: 0,34</p> <p>2. Перевести числа у двійкову</p>	12	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 898 2) $16 \rightarrow 2;10$: F92 3) $8 \rightarrow 2;10$: 341 4) $2 \rightarrow 8;10;16$: 10001111 5) $2 \rightarrow 10;8;16$: 0,1001001 6) $10 \rightarrow 8$: 0,92</p> <p>2. Перевести числа у двійкову</p>

	<p>систему і додати їх у оберненому та доповняльному кодах:-134 і 250</p> <p>3.Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>		<p>систему і додати їх у оберненому та доповняльному кодах:-205 і 50</p> <p>3.Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>
5	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 774</p> <p>2) $16 \rightarrow 2;10$: C7D</p> <p>3) $8 \rightarrow 2;10$: 547</p> <p>4) $2 \rightarrow 8;10;16$: 11100010</p> <p>5) $2 \rightarrow 10;8;16$: 0,11011</p> <p>6) $10 \rightarrow 8$: 0,52</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах:-140 і -26</p> <p>3.Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>	13	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 615</p> <p>2) $16 \rightarrow 2;10$: A9C</p> <p>3) $8 \rightarrow 2;10$: 236</p> <p>4) $2 \rightarrow 8;10;16$: 11000011</p> <p>5) $2 \rightarrow 10;8;16$: 0,1001011</p> <p>6) $10 \rightarrow 8$: 0,82</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах:-117 і 37</p> <p>3.Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>
7	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 932</p> <p>2) $16 \rightarrow 2;10$: EF5</p> <p>3) $8 \rightarrow 2;10$: 157</p> <p>4) $2 \rightarrow 8;10;16$: 10101110</p> <p>5) $2 \rightarrow 10;8;16$: 0,10101</p> <p>6) $10 \rightarrow 8$: 0,87</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах:-17 і -204</p> <p>3.Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>	15	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 911</p> <p>2) $16 \rightarrow 2;10$: 3D1</p> <p>3) $8 \rightarrow 2;10$: 657</p> <p>4) $2 \rightarrow 8;10;16$: 10110001</p> <p>5) $2 \rightarrow 10;8;16$: 0,01011</p> <p>6) $10 \rightarrow 8$: 0,31</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах:-163 і -29</p> <p>3.Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>
8	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 178</p> <p>2) $16 \rightarrow 2;10$: A7E</p> <p>3) $8 \rightarrow 2;10$: 542</p> <p>4) $2 \rightarrow 8;10;16$: 10001010</p> <p>5) $2 \rightarrow 10;8;16$: 0,01011</p> <p>6) $10 \rightarrow 8$: 0,63</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах:230 і -150</p> <p>3.Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>	16	<p>1. Перевести числа:</p> <p>1) $10 \rightarrow 2;8;16$: 826</p> <p>2) $16 \rightarrow 2;10$: C94</p> <p>3) $8 \rightarrow 2;10$: 513</p> <p>4) $2 \rightarrow 8;10;16$: 11000111</p> <p>5) $2 \rightarrow 10;8;16$: 0,110011</p> <p>6) $10 \rightarrow 8$: 0,12</p> <p>2. Перевести числа у двійкову систему і додати їх у оберненому та доповняльному кодах:229 і -61</p> <p>3.Отримати представлення дійсного числа виду $20+N.100+N$ (N-номер варіанту) в пам'яті комп'ютера довжиною 2 байти.</p>

ЛАБОРАТОРНА РОБОТА №3

Тема: Порівняння простих методів сортування

Мета: Реалізувати програмно прості методи сортування. Порівняти їх ефективність на прикладі сортування масивів.

Завдання 1. Скласти блок-схеми та програми для реалізації сортування методами простого вибору, простої вставки, простого обміну. Протестувати роботу програм на прикладі масиву із 50 випадкових дійсних чисел.

Завдання 2. Порівняти час сортування масиву випадкових чисел різними методами. Для цього використати функції бібліотеки `<time.h>`. Відповідні функції викликаються безпосередньо перед початком роботи алгоритму сортування і зразу ж після нього. Різниця складатиме час, затрачений на виконання відповідного алгоритму. Повторити завдання, використавши попередньо відсортований у прямому порядку масив. Повторити завдання, використавши попередньо відсортований у зворотному порядку масив. Зробити висновки.

Завдання 3. Порівняти час сортування різними методами масивів різного характеру (випадкових та відсортованих) із різною кількістю елементів. Зробити висновки про залежність обчислювальних затрат від кількості елементів у кожному випадку.

Отримані результати оформити у вигляді таблиці (приклад на рисунку 3.1), на основі яких зробити висновки про найкращі, найгірші випадки, а також про залежність швидкодії алгоритмів від розмірності масивів. Визначити, який метод у яких випадках є найоптимальнішим.

Прості методи сортування	Випадковий масив, n=500	Відсортований у прямому порядку, n=500	Відсортований у зворотному порядку, n=500	Випадковий масив, n=5000	Відсортований у прямому порядку, n=5000	Відсортований у зворотному порядку, n=5000	Вхідні дані іншої розмірності
1. Метод вибору							
2. Метод вставки							
3. Метод обміну							

Рисунок 3.1 – Приклад оформлення таблиці висновків

Отримання часових характеристик виконання алгоритму

В запропонованому нижче прикладі час вимірюється в секундах, але на сучасних комп'ютерах виконання нескладних алгоритмів складатиме час, менший за 1 секунду, тому порівняти швидкодію буде неможливим.

Для того, щоб знайти час роботи програми, можемо скористатися функцією `clock ()`. Функція `clock ()` повертає значення часу в мілісекундах ($1с = 1000млс$). Причому відлік часу починається з моменту запуску програми. Для пошуку часу роботи фрагмента коду потрібно знайти різницю між кінцевим і початковим часом.

```
#include <time.h>
...
int start_time, end_time;
...
start_time = time(NULL); //безпосередньо перед початком алгоритму
... // сам алгоритм (наприклад, виклик процедури сортування)
int end_time = time(NULL); //після завершення алгоритму
cout << "Час, витрачений на роботу алгоритму: " << end_time -
start_time;

unsigned int start_time = clock(); // початковий час
// фрагмент програми, час виконання якого треба отримати
unsigned int end_time = clock(); // кінцевий час
unsigned int runtime = end_time - start_time;
cout << "runtime = " << runtime /1000.0 << endl;
```

Генерація масиву випадкових чисел

`int rand ()`; - генерує псевдовипадкове число типу `int`.

Псевдовипадкові числа - це числа, які здаються випадковими, але насправді є послідовністю значень, обчислених по певному алгоритму, який приймає за стартове значення параметр, так зване «зерно» (`seed`). Тобто згенеровані функцією `rand ()` числа залежатимуть від значення, яке є зерном в момент її виклику. А зерно завжди встановлюється компілятором значенням 1. Іншими словами, послідовність чисел буде хоч і псевдовипадковою, але завжди однаковою. Проте, необхідно цього уникнути. виправити ситуацію допомагає функція `void srand (unsigned int seed)` - вона встановлює зерно рівним значенню параметра, з яким була викликана. Але значення «зерна» потрібно те зробити випадковим.

Типовий вихід із ситуації – використання в якості зерна функції `time()`, яка поверне поточний час, який відповідно постійно змінюється. Таким чином, маємо:

```
srand(time(NULL));
const long size = 50;
int ar[size];
for (int i = 0; i<size; i++)
{
    ar[i] = rand();
    cout << ar[i] << "\t";
}
```

Приклади створення різних випадкових послідовностей:

!!! rand() % 10 - випадкові числа від 0 до 9.

!!! rand() % 9 + 1 - числа від 1 до 9.

!!! (float)rand()/100 - дійсні числа з двома знаками після коми.

Контрольні запитання:

- 1) Алгоритм сортування методом обміну.
- 2) Алгоритм сортування методом вибору.
- 3) Алгоритм сортування методом вставки.
- 4) Оцінка алгоритмів сортування.
- 5) Модифікації алгоритмів сортування.

ЛАБОРАТОРНА РОБОТА №4

Тема: Алгоритми пошуку елемента та послідовності

Мета: Реалізувати програмно прості та вдосконалені алгоритми пошуку елемента в масиві та слова в тексті. Порівняти їх ефективність на прикладі.

Завдання 1. Реалізуйте програму, яка буде знаходити індекс елементу масиву за заданим ключем двома методами – методом лінійного пошуку та бінарного. Порівняйте швидкодію алгоритмів, зробіть висновки про найкращі та найгірші випадки вхідних даних.

Завдання 2. Реалізуйте програму, яка буде знаходити входження слова в тексті двома методами – прямим алгоритмом пошуку послідовності та КМП-пошуком. Результатом роботи алгоритмів повинен бути індекс першого символу, з якого починається співпадіння. Порівняйте швидкодію алгоритмів, зробіть висновки про найкращі та найгірші випадки вхідних даних.

Завдання 3(додаткове). Реалізуйте програму, яка буде знаходити входження слова в тексті з використанням алгоритму БМ-пошуку. Порівняйте швидкодію алгоритму з алгоритмами, реалізованими в Завданні 2, зробіть висновки про найкращі та найгірші випадки вхідних даних.

В звіт включити лістинги алгоритмів, тестові приклади, висновки щодо швидкодії алгоритмів в різних випадках.

Контрольні питання:

1. Що таке пошук?
2. Що називаються ключем пошуку?
3. Які відомі методи пошуку?
4. Який алгоритм пошуку є найбільш ефективним?
5. Чим відрізняються пошук в масиві від пошуку в списку?
6. У чому полягає метод лінійного пошуку?
7. У чому полягає метод бінарного пошуку?
8. Прямий пошук у рядку?
9. У чому полягає метод КМП пошуку?
10. Основна ідея алгоритму пошуку БМ?

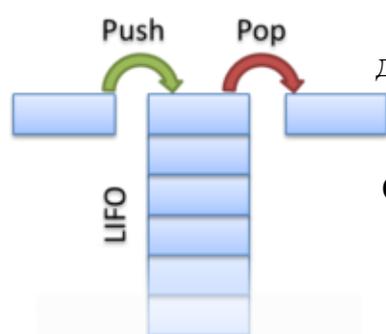
ЛАБОРАТОРНА РОБОТА № 5

Тема: Робота з лінійними зв'язаними списками. Структури даних стек та черга.

Мета роботи: Отримати навички роботи зі стеком та чергою, реалізованими у вигляді зв'язного лінійного списку.

Теоретичні відомості:

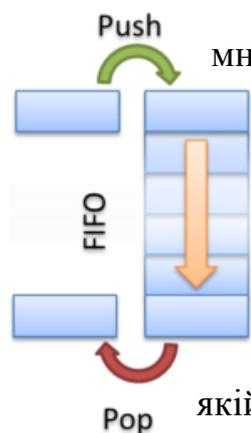
Стеки та черги — це динамічні структури даних, які реалізуються за допомогою зв'язних лінійних списків.



Першим зі **стеку (stack)** видаляється елемент, який був туди доданий останнім: в стеку реалізується стратегія "останнім зайшов — першим вийшов" (last-in, first-out — LIFO).

Отже, всі операції (наприклад, видалення елемента) в стеку можна проводити тільки з одним елементом, що знаходиться на вершині стеку і був введений в стек останнім.

Стек можна представити, як сукупність однотипних елементів, в якій ми маємо доступ тільки до верхнього елемента.



У **черзі (queue)** завжди видаляється елемент, який міститься у множині довше від інших: в черзі реалізується стратегія "першим зайшов – першим вийшов" (first-in, first-out — FIFO).

У черги є голова (англ. head) та хвіст (англ. tail). Елемент, що додається до черги, опиняється в її хвості. Елемент, що видаляється з черги, знаходиться в її голові.

Чергу можна представити, як сукупність однотипних елементів, в якій маємо доступ до кінця черги при додаванні елементів та до початку черги при взятті елементів.

Завдання до виконання

Варіант 1. Робота з структурою даних «стек»:

Розробити програму роботи зі стеком, яка реалізує наступні операції:

- додавання елементів у стек
- видалення елементів зі стеку
- відображення поточного стану стеку (наприклад $3 \Rightarrow 5 \Rightarrow 2 \Rightarrow 7$)
- отримання розміру стеку
- пошук елемента в стеку

Реалізувати стек на основі однозв'язних списків (з використанням структур та покажчиків). Приклад реалізації такого списку та функції додавання елементів подано нижче, інші дії необхідно реалізувати самостійно.

В програмі повинно бути реалізоване меню для вибору дії з стеком. Кожна з операцій повинна бути реалізована в окремій функції.

Варіант 2. Робота з структурою даних «черга»:

Розробити програму роботи зі стеком, яка реалізує наступні операції:

- додавання елементів у чергу
- видалення елементів з черги
- відображення поточного стану черги (наприклад $3 \Rightarrow 5 \Rightarrow 2 \Rightarrow 7$)
- отримання розміру черги
- пошук елемента в черзі

Реалізувати чергу на основі однозв'язних списків (з використанням структур та покажчиків). Приклад реалізації такого списку та функції додавання елементів подано нижче, інші дії необхідно реалізувати самостійно.

В програмі повинно бути реалізоване меню для вибору дії з чергою. Кожна з операцій повинна бути реалізована в окремій функції.

Додаткові завдання:

1. Створити і вивести два стеки – з парними та непарними числами серед введених.
2. . Реалізувати основні функції по роботі з деками, в яких записані цілочисельні дані. Реалізувати функцію контролю (виводу) поточного стану дека. Передбачити недопустимість помилок при некоректних діях (видалення елемента з порожнього деку).

Приклад реалізації однозв'язного списку:

```
struct Node //створення структури вузла
{
    int value; // певна інформаційна частина
    Node * next; // вказівник на наступну структуру-вузол у списку
};

Node * head = NULL; // вказівник на голову списку, спочатку він
нікуди не вказує, бо список порожній

void addToEnd(int v) // Реалізація функції додавання у кінець списку
{
    Node * n;
    if (!head)
    {
        head = new Node;
        head->value = v;
        head->next = NULL;
        return;
    }
    else
    {
        n = head;
        while (n->next)
            n = n->next;
        Node * newNode = new Node;
        newNode->value = v;
        newNode->next = NULL;
        n->next = newNode;
        return;
    }
}

void addToBegin(int v) // Реалізація функції додавання у початок
списку
{
    Node * n = new Node;
    n->value = v;
    n->next = head;
    head = n;
}
```

Контрольні запитання:

1. Що представляє собою «список» як структура даних?
2. Які особливості організації даних у структурі «список»?
3. Які основні функції по роботі зі списками?
4. Які структурні елементи обов'язково містить список, а які – можуть бути додатковими?
5. В чому переваги та недоліки організації списків за допомогою масивів?
6. В чому переваги та недоліки організації списків за допомогою вказівників?
7. Наведіть порівняльну характеристику способів організації списків.
8. Наведіть приклади доцільності використання масивів для організації списку.
9. Наведіть приклади доцільності використання списку на основі вказівників.
10. У чому особливість кільцевих списків?
11. Що таке стек?
12. Які основні функції по роботі зі стеком?
13. Чи потрібно стек організувати на основі двозв'язних списків?
14. Що таке черга?
15. Які основні функції по роботі з чергою?
16. Чим відрізняються черги від простих списків?
17. Чим відрізняються стек і черга?
18. Які структурні елементи обов'язково містить стек?
19. Які структурні елементи обов'язково містить черга?
20. На основі яких списків доцільно організувати черги?
21. Наведіть порівняльну характеристику стеків і звичайних однозв'язних списків.
22. Наведіть порівняльну характеристику черг і одно- та двозв'язних списків.

ЛАБОРАТОРНА РОБОТА №6

Тема: Робота з бінарним деревом пошуку

Приклад реалізації дерева:

```
//Оголошення структури дерева
struct node
{
    int info; //Інформаційне поле
    node *l, *r;// Ліве і праве піддерева
};
node * tree=NULL; //корінь дерева

/*функція додавання елемента в бінарне дерево*/
void push(int a,node **t)
{
    if ((*t)==NULL) //якщо дерево порожнє
    {
        (*t)=new node; //виділяємо пам'ять
        (*t)->info=a; //Інформаційну частину заповнюємо значенням
аргументу
        (*t)->l=(*t)->r=NULL; //Ліве і праве піддерева - NULL
        return;
    }
    //якщо дерево не порожнє
    if (a>(*t)->info) push(a,&(*t)->r); //Якщо аргумент а >поточний
елемент,
                                     то додавати будемо в праве піддерево
    else push(a,&(*t)->l); //янакше, якщо а<поточний елемент,
                                     то додаємо вліво
}

/*функція відображення дерева на екрані*/
void print (node *t,int u)
{
    if (t==NULL) return; //якщо дерево пусте, то виходимо з функції
    else
    {
        print(t->l,++u); //через рекурсивний виклик йдемо в ліве
піддерево
        for (int i=0;i<u;++i) cout<<"|";
        cout<< t->info << endl; //і виводимо елемент
        u--;
    }
    print(t->r,++u); //через рекурсивний виклик йдемо в праве піддерево
}

void main ()
{
    int n; //кількість елементів
    int s; //число, що додається до дерева
    cout<<"введіть кількість елементів ";
    cin>>n;
```

```

for (int i=0;i<n;++i)
{
cout<<"введіть число ";
cin>>s;
push(s,&tree); //кожен введений елемент додаємо в дерево
}
cout<<"ваше дерево\n";
print(tree,0);
getch();
}

```

Завдання: Реалізуйте програму для роботи з бінарним деревом пошуку. В програмах повинно бути реалізоване меню для вибору дії (додавання елемента з вказанням значення, видалення елемента та виведення дерева на екран). Кожна з опцій реалізовується окремою функцією.

Додаткове завдання: Написати програму, в якій дані вашого варіанту структури записуються в дерева (використати три поля для вузла – текстове дане та два числові. Наприклад, вузол дерева містить таку корисну інформацію: прізвище студента, рік народження, оцінка. Ввести з клавіатури декілька "студентів" у двійкове дерево, організоване за порядком текстового поля. Вивести отримане дерево.

Питання для контролю та самоконтролю?

1. Що таке «дерево» як структура даних?
2. Що таке «корінь дерева»? Глибина дерева? Висота дерева
3. Які основні властивості дерева?
4. Яке дерево називають n-арним? бінарним?
5. Чи можна з n-арного дерева побудувати бінарне? Якщо можна, то як?
6. Які основні функції по роботі з деревами?
7. Яким чином при роботі зі структурою «дерево» використовуються рекурсивні алгоритми?
8. Які особливості організації вузла дерева?
9. Яким чином дерева використовують для впорядкування даних?

Лабораторна робота № 7

Тема: Методи пошуку в графах. Пошук у ширину та глибину. Алгоритми на графах.

Завдання до лабораторної роботи.

Варіант1.

1. Реалізуйте алгоритм пошуку в ширину на прикладі графа не менше ніж з 6-ти вершин.
2. Реалізуйте алгоритм Прима побудови мінімального остовного дерева.

Варіант2.

1. Реалізуйте алгоритм пошуку в глибину на прикладі графа не менше ніж з 6-ти вершин.
2. Реалізуйте алгоритм Дейкстри для пошуку найкоротшого шляху від заданої до заданої вершини.

Додаткове завдання. Реалізувати будь-який з наведених варіантів на вибір студента з використанням списків суміжності.

Приклади реалізації подібних завдань.

Пошук в ширину:

```
#include "stdafx.h"
#include <iostream>
using namespace std;

const int n=4;
int i, j;
int GM[n][n] = {{0, 1, 1, 0},           //матриця суміжності графа
                {0, 0, 1, 1},
                {1, 0, 0, 1},
                {0, 1, 0, 0}};

void BFS(bool *visited, int unit) //пошук в ширину
{
    int *queue=new int[n];
    int count, head;
    for (i=0; i<n; i++) queue[i]=0;
    count=0; head=0;
    queue[count++]=unit;
    visited[unit]=true;
    while (head<count)
    {
        unit=queue[head++];
        cout<<unit+1<<" ";
        for (i=0; i<n; i++)
```

```

        if (GM[unit][i] && !visited[i])
        {
            queue[count++]=i;
            visited[i]=true;}}
    delete []queue;
}
void main(){ //головна функція
    setlocale(LC_ALL, "Rus");
    int start;
    cout<<"Стартова вершина >> "; cin>>start;
    bool *visited=new bool[n];
    cout<<"Матриця суміжності графу: "<<endl;
    for (i=0; i<n; i++)
    {
        visited[i]=false;
        for (j=0; j<n; j++)
            cout<<" "<<GM[i][j];
        cout<<endl;}
    cout<<"Порядок обходу: ";
    BFS(visited, start-1);
    delete []visited;
    system("pause>>void");
}

```

Пошук в глибину:

```

#include "stdafx.h"
#include <iostream>
using namespace std;
const int n=5;
int i, j;
bool *visited=new bool[n];
int graph[n][n] = { //матриця суміжності графа

    {0, 1, 0, 0, 1},
    {1, 0, 1, 1, 0},
    {0, 1, 0, 0, 1},
    {0, 1, 0, 0, 1},
    {1, 0, 1, 1, 0}};
void DFS(int st) //пошук в глибину
{
    int r;
    cout<<st+1<<" ";
    visited[st]=true;
    for (r=0; r<=n; r++)
        if ((graph[st][r]!=0) && (!visited[r]))
            DFS(r);
}
void main() //головна функція
{
    setlocale(LC_ALL, "Rus");
    int start;
    cout<<"Матриця суміжності графу: "<<endl;
    for (i=0; i<n; i++)

```

```

        {
            visited[i]=false;
            for (j=0; j<n; j++)
                cout<<" "<<graph[i][j];
            cout<<endl;
        }
    cout<<"Стартова вершина >> "; cin>>start;
    bool *vis=new bool[n]; //масив відвіданих вершин
    cout<<"Порядок обходу: ";
    DFS(start-1);
    delete []visited;
    system("pause>>void");
}

```

Питання для контролю та самоконтролю.

1. Що називається графом?
2. Що таке повний граф? Зв'язний граф? Орієнтований граф?
3. Як формується матриця суміжності?
4. Що називається списком суміжності?
5. Як здійснити обхід графа?
6. Як виконати пошук у ширину?
7. Які структури необхідні для реалізації пошуку у ширину?
8. Як виконати пошук у глибину?
9. Які структури необхідні для реалізації пошуку у глибину?
10. Аналіз алгоритмів?
11. Що таке остовне дерево?
12. Які є способи побудови остовних дерев ?
13. Як реалізувати зафарбовування вершин?
14. Що таке мінімальна вартість остовного дерева?
15. Алгоритм Пріма, основні кроки?
16. Алгоритм Крускала, основні кроки?
17. Як знайти найкоротший шлях у графі?
18. Алгоритм Дейкстри, основні кроки?
19. Алгоритм Флойда, основна ідея?
20. Аналіз алгоритмів?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы. – М.: Изд. Дом "Вильямс", 2003. – 384 с.
2. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.
3. Вирт Н. Алгоритмы и структуры данных. – М.: Мир, 1989. – 360 с.
4. Далека В.Д., Деревянко А.С., Кравец О.Г., Тимановская Л.Е. Модели и структуры данных: Учебное пособие. – Харьков: ХГПУ, 2000. – 241с.
5. Кнут Д. Искусство программирования. Т.1. Основные алгоритмы. – М.: Вильямс, 2002. – 720 с.; Т.2. Получисленные алгоритмы. – М.: Вильямс, 2004. – 832 с.; Т.3. Сортировка и поиск. – М.: Вильямс, 2005 – 824 с.
6. Ковалюк Т.В. Основи програмування. – К.: Видавнича група ВНУ, 2005. – 384 с.
7. Седжвик Р. Фундаментальные алгоритмы на С++. Алгоритмы на графах. – СПб: ООО «ДиаСофтЮП», 2002. – 496 с.
8. Седжвик Р. Фундаментальные алгоритмы на С++. Анализ / Структуры данных / Сортировка / Поиск. – К.: Изд-во «ДиаСофт», 2001. – 688 с.
9. Ставровський А.Б., Карнаух Т.О. Програмування. Перші кроки. – М.: Вид. дім "Вільямс", 2005. – 400 с.
10. В.Є. Величко, М.М. Рубан, В.П. Батунина, С.Є. Устінов. Олімпіадні задачі з інформатики: Розв'язання задач II етапу Всеукраїнської олімпіади з інформатики – 2007, 2008 рр.. – Слов'янськ, 2009. – 34 с.
11. Караванова Т.П. Інформатика: основи алгоритмізації та програмування: 777 задач, з рекомендаціями та прикладами К.: Генеза, 2009.- 285 с.
12. В.М. Ільман. Алгоритми, дані і структури. Навч. посіб. /О.П. Іванов, Л.О. Панік. Дніпропет. нац. ун-т залізн. трансп.ім. акад. В. Лазаряна. – Дніпро, 2019. – 134 с.
13. Крєневич А.П. Алгоритми і структури даних. Підручник. – К.: ВПЦ "Київський Університет", 2021. – 200 с.
14. Томас Г. Кормен, Чарлз Е. Лейзерсон, Роналд Л. Рівест, Кліфорд Стайн Вступ до алгоритмів. — К. : [K. I. C.](http://www.kit.com.ua), 2019. — 1288 с.
15. Алгоритми і структури даних: навч. посіб. / Т. О. Коротеєва ; М-во освіти і науки України, Нац. ун-т «Львів. політехніка». — Львів: Вид-во Львів. політехніки, 2014. — 280 с.

Додаткові джерела

1. Курс CS50 : https://courses.prometheus.org.ua/courses/course-v1:Prometheus+CS50+2019_T1
2. Розробка та аналіз алгоритмів: https://courses.prometheus.org.ua/courses/KPI/Algorithms101/2015_Spring/course/

3. Робота з деревами: <http://cppstudio.com/uk/293/cat/>
<https://www.youtube.com/watch?v=qBFzNW0ALxQ>
4. Візуалізація алгоритмів КМП та БМ:
<http://jovilab.sinaapp.com/visualization/algorithms/strings/kmp>
<http://jovilab.sinaapp.com/visualization/algorithms/strings/boyer-moore-horspool>
5. Візуалізація деяких алгоритмів на графах:
<https://sites.google.com/site/chnodatstruct/home/vizualizacia-deakih-algoritmiv>
<https://graphonline.ru/>