МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

О. О. Петрова,
Г. В. Солодовник

АЛГОРИТМІЧНІ ЗАДАЧІ ТА ЇХ ВИРІШЕННЯ

НАВЧАЛЬНИЙ ПОСІБНИК

Харків
ХНУМГ ім. О. М. Бекетова
2021

УДК 510.5 (075.8)
П30

Автори:

Петрова Олена Олександрівна, кандидат технічних наук, доцент кафедри комп'ютерних наук та інформаційних технологій Харківського національного університету міського господарства імені О. М. Бекетова;

Солодовник Ганна Валеріївна, кандидат технічних наук, доцент кафедри комп'ютерних наук та інформаційних технологій Харківського національного університету будівництва та архітектури

Рецензенти:

Чуб Ігор Андрійович, доктор технічних наук, професор, начальник кафедри ПІНП Національного університету цивільного захисту України;

Костенко Олександр Борисович, кандидат фізико-математичних наук, доцент кафедри комп'ютерних наук та інформаційних технологій Харківського національного університету міського господарства імені О. М. Бекетова

Рекомендовано до друку Вченою радою ХНУМГ ім. О. М. Бекетова, протокол № 12 від 01.07.2020.

Петрова О. О.

П30 Алгоритмічні задачі та їх вирішення : навч. посібник / О. О. Петрова, Г. В. Солодовник ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2021. – 105 с.

Розроблений навчальний посібник містить матеріал щодо засад алгоритмізації, призначення, властивостей, способів опису алгоритмів, побудови алгоритмів лінійних, розгалужених та циклічних процесів. Матеріали визначених тем присвячено алгоритмізації одномірних та двомірних масивів, розгляду фундаментальних алгоритмів сортування. У цьому виданні наводиться докладне пояснення щодо побудови алгоритмів, описи виконання типових завдань та варіанти індивідуальних завдань до практичних занять та самостійного вивчення тем.

Призначено для студентів спеціальностей 122 – Комп'ютерні науки, 126 – Інформаційні системи та технології, а також усіх тих, хто цікавиться цими питаннями.

УДК 510.5 (075.8)

ЗМІСТ

ВСТУП

Під час розв'язання науково-технічних задач велика увага приділяється розробці алгоритму їх розв'язання. Незважаючи на зусилля дослідників, єдиного вичерпного й чіткого визначення поняття алгоритму не існує.

Варіанти вербального визначення алгоритму, що вже існують, належать російським вченим А. М. Колмогорову і А. А. Маркову [1–2]:

Визначення 1 (за Колмогоровим): *Алгоритм* – це будь-яка система обчислень, що виконується за чітко визначеними правилами, яка після певного числа кроків призводить до розв'язання поставленої задачі.

Визначення 2 (за Марковим): *Алгоритм* – це чіткий припис, що визначає обчислювальний процес, який іде від первинних даних, що варіюються, до результату, який відшукується.

Зазначимо, що різні визначення алгоритму, в явній або неявній формах, визначають такі вимоги [3]:

1) алгоритм повинен містити скінченну кількість приписів, що виконуються елементарно, тобто задовольняти вимогам скінченності запису;

2) алгоритм повинен виконувати скінченну кількість кроків під час розв'язання задачі, тобто задовольняти вимогам скінченності дій;

3) алгоритм повинен бути єдиним для всіх допустимих первинних даних, тобто задовольняти вимогам універсальності;

4) алгоритм повинен призводити до отримання вірного розв'язку задачі, що ставиться, тобто задовольняти вимогам правильності.

Розглянемо основні принципи, за якими будуються алгоритми:

1. Алгоритм застосовується щодо первинних даних і генерує результати. У технічних термінах це означає, що алгоритм має входи й виходи. Крім того, в ході роботи алгоритму з'являються проміжні результати, які використовуються в подальшому. Отже, кожен алгоритм має справу з вхідними, проміжними та вихідними даними.

2. Дані для свого розміщення вимагають застосування пам'яті, яка є однорідною та дискретною, тобто складається з однакових клітинок, до того ж кожна комірка може містити один символ алфавіту даних.

3. Робота алгоритму складається з окремих елементарних кроків, до того ж множина кроків, з яких складено алгоритм є скінченною.

4. Послідовність кроків алгоритму є детермінованою, тобто після кожного кроку вказується, який крок потрібно виконувати далі, або дається команда зупинки, після чого робота алгоритму вважається закінченою.

5. Природною для алгоритму є вимога результативності, тобто зупинення після кінцевого числа кроків із зазначенням того, що вважати результатом.

Потрібно розрізняти:

– опис алгоритму (реалізовану програму);

– механізм реалізації, що включає засоби пуску, зупинення, реалізації елементарних кроків, видачі результатів та забезпечення управління перебігом обчислення;

– процес реалізації алгоритму, тобто послідовність кроків, яка утвориться в результаті застосування алгоритму до конкретних даних [3].

Для створення алгоритму необхідно знати:

– яку роботу повинен виконувати алгоритм;

– якими повинні бути вхідні дані;

– якими повинні бути вихідні дані.

Алгоритми як логіко-математичні засоби відображають певні компоненти та тенденції людської діяльності. Залежно від мети, початкових умов задачі, шляхів її розв'язання та визначення дій виконавця існують такі види алгоритмів:

1. Механічні алгоритми (детерміновані або жорсткі), що задають певні дії, розташовуючи їх у єдиній та достовірній послідовності й забезпечуючи в такий спосіб однозначний необхідний або бажаний результат у разі, якщо виконуються умови процесу або задачі, для яких розроблено алгоритм.

2. Гнучкі алгоритми, наприклад стохастичні, тобто ймовірнісні та евристичні. Імовірнісний (стохастичний) алгоритм надає програму для розв'язання задачі декількома шляхами або способами, що призводять до вірогідного досягнення результату. Евристичний алгоритм – це алгоритм, що використовує різні розумні міркування без чітких обґрунтувань.

Це видання містить необхідний теоретичний матеріал та ілюстровані приклади складання алгоритмів.

Після вивчення відповідної теми студенти зможуть самостійно перевірити свої знання за допомогою контрольних питань, які наводяться в кінці кожного розділу та вдосконалити навички роботи під час виконання індивідуальних завдань варіанти, яких наведено в кожному практичному занятті.

Навчальний посібник дасть змогу студентам:

– ознайомитися з формами подання алгоритмів та зі структурою алгоритмів;

– ознайомитися з основними характеристиками алгоритму: алфавітом даних та формою їх подання, пам'яттю та розміщенням у ній елементів;

– розробити найефективніший алгоритм розв'язання поставленої задачі з урахуванням обчислювальних процесів, необхідних для роботи алгоритму;

– зрозуміти, що з точки зору сучасної практики алгоритм є реалізованою програмою, а критерієм алгоритмічності процесу є можливість його запрограмувати;

– ознайомитися з фундаментальними алгоритмами.

1 ФОРМАЛІЗАЦІЯ ТА АЛГОРИТМІЗАЦІЯ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Під час розв'язання задач визначеного спрямування: математичних, інженерних, економічних та інших, розробник повинен:

1) сформулювати умови задачі;
2) виконати математичний опис задачі;
3) обрати та обґрунтувати метод розв'язання задачі;
4) провести алгоритмізацію обчислювального процесу;
5) скласти програму;
6) налагодити програму;
7) розв'язати задачу та проаналізувати результати.

Найбільш важливим та відповідальним із цих кроків є алгоритмізація.

Алгоритм – це заданий певною мовою скінченний припис, що задає скінченну послідовність виконуваних елементарних операцій для розв'язання задачі, що є загальним для класу можливих первинних даних.

Нехай D – область (множина) первинних даних задачі, а R – множина можливих результатів, тоді можна говорити, що алгоритм здійснює відображення $D \rightarrow R$.

Алгоритмами, наприклад, є правила додавання, множення, розв'язання алгебраїчних рівнянь, множення матриць тощо. Слово «алгоритм» походить від латинського algoritmi, що є латинською трансляцією арабського імені хорезмійського математика IX століття Абу́ Абдулла́х (або Абу Джафар) Мухаммад ибн Муса́ аль-Хорезмі. Завдяки латинському перекладу трактату аль-Хорезмі європейці в XII столітті ознайомилося з позиційною системою числення, а в середньовічній Європі алгоритмом називалася десяткова позиційна система числення та правила розрахунків за її допомогою.

Для запису послідовності дій, описаних алгоритмом у комп'ютерах, потрібна формалізована мова – мова програмування, а сам запис алгоритму такою мовою називають програмою.

Алгоритм повинен мати такі властивості:

1) *зрозумілість* – виконувач алгоритму повинен знати, як його виконувати;

2) *дискретність* – алгоритм повинен відображати процес розв'язання задачі як послідовне виконання простих (або раніше визначених) кроків (етапів);

3) *результативність* – можливість отримання результату після виконання кінцевої кількості операцій;

4) *визначеність* – збіг отриманих результатів незалежно від користувача та технічних засобів, що використовуються;

5) *масовість* – можливість використання алгоритму до цілого класу однотипних задач, що розрізняються конкретними значеннями первинних даних [1–3].

Існують такі типи алгоритмів:

1) *лінійний алгоритм* – це алгоритм, дії якого виконуються послідовно одна за іншою;

2) *розгалужений алгоритм* – це алгоритм, дії якого виконуються залежно від виконання (або невиконання) певної умови (питання, на яке можна відповісти тільки «так» або «ні»);

3) *циклічний алгоритм* – це алгоритм, дії якого повторюються.

До способів опису алгоритмів належать:

1) словесно-формульний, за якого алгоритм записується у вигляді тексту з формулами відповідно до пунктів, що визначають послідовність дій;

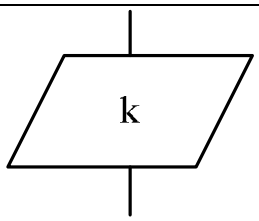
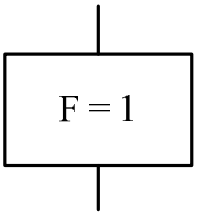
2) структурний або блок-схемний, за якого алгоритм зображується геометричними фігурами (блоками), пов'язаними між собою відповідно до управління лініями (напрямками потоків) зі стрілками; в блоках записується послідовність дій;

3) із використанням граф-схем;

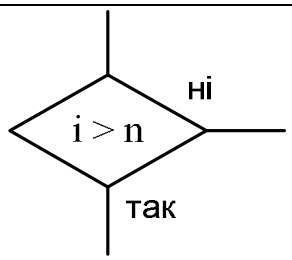
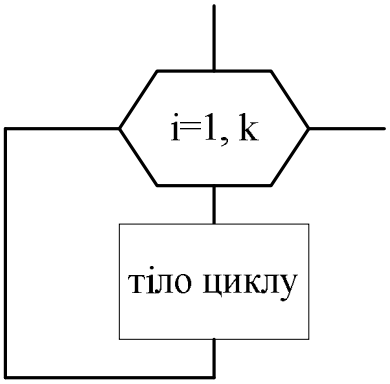
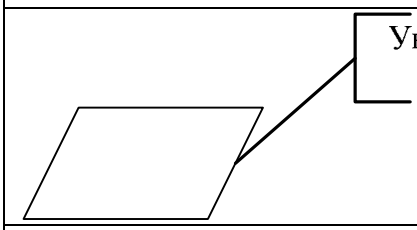
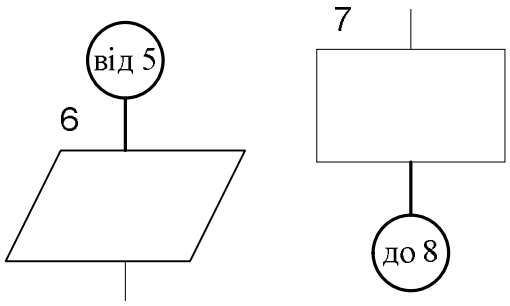
4) із використанням мереж Петрі.

Блок-схемний спосіб опису алгоритмів передбачає записи дій у блоках різного призначення. Призначення блоку визначається його формою та змістом. У таблиці 1.1 наведено блоки та їхнє призначення.

Таблиця 1.1 – Призначення блоків структурного опису алгоритмів

Вигляд блоку	Призначення блоку
1	2
	Блок початку. Має лише один вихідний потік, завжди розташовується на початку блок-схеми алгоритму
	Блок кінця. Має лише один вхідний потік, завжди розташовується на кінці блок-схеми алгоритму
	Блок уведення/виведення даних. Призначений для введення/виведення даних користувачем із клавіатури або з інших файлів. Має один вхідний та один вихідний потоки
	Блок привласнення. Призначений для виконання арифметичних та інших операцій (іноді це може бути виконання заздалегідь описаної функції або процедури). Має один вхідний та один вихідний потоки

Продовження таблиці 1.1

1	2
	Блок умови. Призначений для перевірки умови та визначення подальшого напряму обчислюваного процесу згідно з цією умовою. Має один вхідний та два вихідних потоки («*так*» або «*ні*»)
	Блок циклу. Призначений для організації циклів, містить значення лічильника циклу: *змінна_лічильник = початкове_значення*, *кінцеве_значення*, *крок_зміни_значення*. Якщо останнє значення відсутнє, то крок зміни дорівнює одиниці. Має один вхідний потік (зверху), два вихідних: до блоків тіла циклу (знизу) та до блоків, які виконуються після завершення циклу (праворуч), потік зациклення (ліворуч), за яким цикл повертається для виконання наступного проходження у випадку якщо циклічний процес ще не завершено
	Коментар. Призначений для внесення на блок-схему додаткової інформації для пояснень тих або інших дій
	Позначки переносів блок-схеми. Використовуються, якщо блок-схема не поміщається на одному аркуші. Може крім номера блоку містити номер сторінки, на якій даний блок розташовано

1.1 Алгоритмізація лінійного обчислювального процесу

Лінійним прийнято називати обчислювальний процес, операції якого виконуються послідовно, в порядку їх запису. Кожна операція є самостійною й не залежить від будь-якої вимоги. На схемі блоки, що відображають ці операції, розташовані у лінійній послідовності.

Лінійна структура – це найпростіший тип алгоритму, в процесі виконання якого відтворюється послідовне виконання блоків алгоритму. Об’єктом опису в схемах алгоритмів є константи, змінні величини, арифметичні та логічні вирази.

Константа – це незмінна величина визначеного типу.

Змінна – це елемент алгоритму (програми), що має ідентифікатор (унікальне ім'я), призначений для зберігання, корекції та передавання даних.

Арифметичний вираз – це сукупність констант, змінних та функцій, пов'язаних між собою знаками арифметичних операцій. Послідовність виконання операцій у арифметичному виразі така:

1) операції в дужках;
2) обчислення функцій;
3) піднесення до ступеня;
4) множення та ділення;
5) додавання та віднімання.

Операції з однаковим пріоритетом виконуються зліва направо. Під час запису арифметичних виразів у алгоритмі потрібно дотримуватись таких правил:

1) усі знаки потрібно записувати чітко, зокрема й знак множення;
2) знаки операцій не можна записувати підряд, між ними обов'язково потрібно використовувати операнди;
3) під час послідовного піднесення до ступеня порядок виконання операцій потрібно вказувати за допомогою дужок;
4) кількість дужок, що відкриваються та що закриваються, повинна збігатися;
5) після назви функції потрібно вказувати її аргументи у дужках, наприклад, SIN(x).

Лінійний обчислювальний процес характеризується тим, що кроки, на які він розбивається, виконуються послідовно в тому порядку, в якому вони подані.

Під час запису арифметичного виразу використовуються стандартні функції (табл. 1.2).

Таблиця 1.2 – Математичні функції

Функція	Позначення функції	Пояснення функції
sin x	sin(x)	Функція синуса
cos x	cos(x)	Функція косинуса
tg x	tan(x)	Функція тангенса
lg x	log(x)	Функція натурального логарифма
|x|	abs(x)	Функція модуля
e^x	exp(x)	Експоненціальна функція
$\sqrt{x}$	sqrt(x)	Функція квадратного кореня

За допомогою лінійних алгоритмів вирішуються прості задачі, в яких число дій дорівнює числу блоків схеми алгоритма.

1.2 Алгоритмізація розгалуженого обчислювального процесу

Розгалуженим називають алгоритм, у якому обрання дії залежить від виконання певних умов та значень вхідних даних або проміжних результатів.

Алгоритми розгалужених обчислювальних процесів становлять структури, що мають декілька варіантів обчислень. Кожний варіант поданий окремою гілкою, обрання якої залежить від умови або отриманих проміжних результатів.

Блок-схема алгоритму розгалуженого обчислюваного процесу містити блок умови, в якому перевіряється логічна умова.

Логічні умови можуть бути об'єднані в логічні вирази за допомогою таких операцій: логічного додавання, логічного множення, логічного «ні».

Існують певні типи логічних відносин, а саме: відношення еквівалентності, відношення порядку.

1.3 Алгоритмізація циклічного обчислювального процесу

Циклічний алгоритм – це алгоритм, що передбачає багаторазове повторення однієї й тієї самої дії (операції) над новими первинними даними. До циклічних алгоритмів зводиться більшість методів обчислень, перебору варіантів. Цикл програми – це послідовність команд (тіло циклу), які можуть виконуватися багаторазово (для нових первинних даних) доти, доки не буде задовольнятися певна умова.

Вкладеним називається цикл, що містить у собі один або декілька інших циклів. Цикл, що охоплює інші цикли, називається зовнішнім, а інші – внутрішніми.

Кожен цикл окремо організований так само, як і простий цикл. Вкладеними в циклічних процесах можуть бути не лише інші цикли, але й розгалуження та фрагменти лінійного типу.

1.4 Алгоритмізація роботи з одновимірними масивами

Під час обробки значних обсягів даних використання одних лише базових типів змінних не достатньо. У мовах програмування існує спосіб, який надає можливість зручно розміщувати в пам'яті велику кількість необхідної інформації – це спосіб використання масивів.

Масив – це набір змінних, які мають одне й те саме базове ім'я, але відрізняються одна від одної числовою ознакою (індексом).

Масиви можуть мати ті самі типи даних, які мають прості змінні. Ім'я масиву вказується одразу ж за ключовим словом, яке визначає тип його елементів. Дужки після імені вказують на те, що це масив, а число в дужках – на кількість елементів масиву. Відрахунок елементів масиву можна починати з 0, або з 1. Окремий елемент масиву визначається за допомогою його номера, який називається індексом.

Для того щоб звернутися до елемента масиву, можна використовувати константи, змінні та вирази цілого типу. Елементи масиву розміщуються в

пам’яті послідовно один за одним. Кожному елементу відводиться в пам’яті стільки саме місця, скільки й простій змінній такого самого типу.

Елементам оголошеного масиву можна надати початкових значень. При цьому список значень потрібно розташовувати в фігурних дужках, а самі значення розділяти комами: *int a[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30}.*

У процесі компіляції під масив відводиться пам’ять у розмірі, який дорівнює кількості елементів, помноженій на розмір пам’яті, яка відводиться для окремого елемента.

1.5 Алгоритмізація роботи з двовимірними масивами

Двовимірний масив – це масив масивів. Розмір двовимірного масиву задається після імені масиву в дужках, наприклад: *int a[2] [2].*

Уводити числові значення елементів масиву можна:

– безпосередньо з клавіатури;

– записуючи числові значення після опису масиву: *Int a[2] [2] = 1, 2, 4, 8.*

Запис масиву з двох рядків та чотирьох стовпчиків виглядає так:

int a[2] [4] = 11, 12, 13, 14,
21, 22, 23, 24.

1.6 Алгоритмізація процесу сортування

Під сортуванням, зазвичай, розуміють процес перестановки об’єктів конкретної множини у визначеному порядку. Мета сортування – полегшити відшукування елементів у множині, яка сортується. У цьому розумінні елементи сортування присутні майже в усіх задачах.

Із сортуванням пов’язано багато фундаментальних прийомів побудови алгоритмів. Розглянемо сортування масивів [4–5].

Нехай задано елементи:

$$a_1, a_2, ..., a_n.$$

Процес сортування означає здійснення перестановки даних елементів у такому порядку:

$$a_{k_1}, a_{k_2}, ..., a_{k_n},$$

тобто за заданої *функції впорядкування f* справедливим є відношення

$$f(a_{k_1}) \le f(a_{k_2}) \le ... \le f(a_{k_n}).$$

Зазвичай, функція впорядкування не обчислюється за якимось спеціальним правилом, вона міститься в кожному елементі у вигляді явної компоненти (поля), значення якої називається ключем елемента.

Основною вимогою до методів сортування масивів є економне використання пам’яті, а отже, під час обрання методу сортування потрібно керуватися критерієм економії пам’яті. У зв’язку з цим класифікацію алгоритмів будемо проводити відповідно до їхньої ефективності, тобто економії часу або швидкодії.

Зручну міру ефективності отримуємо за допомогою підрахунку чисел C (кількості необхідних порівнянь) та M (кількості пересилань елементів). Ці числа визначаються деякими функціями від числа n (кількості елементів), що сортуються. Ефективніші алгоритми сортування потребують порядку $n \cdot log\ n$ порівнянь, але спочатку розглянемо кілька нескладних та очевидних способів сортування, які мають назву простих методів і потребують порядку n^2-порівнянь.

Методи для сортування елементів *in situ* можна розбити на три основних класи залежно від прийому, закладеного в основу методу: сортування методом включень, сортування методом вибору, сортування методом обміну [2–4].

Контрольні питання

1. Дати визначення поняття «алгоритм».
2. Охарактеризувати основні принципи, за якими будуються алгоритми.
3. Описати етапи розв’язання задачі.
4. Охарактеризувати властивості алгоритмів.
5. Навести способи опису алгоритмів.
6. Для чого призначені блоки структурного опису алгоритмів?
7. Охарактеризувати типи алгоритмів.
8. Охарактеризувати лінійний обчислювальний процес.
9. Перелічити складники арифметичного виразу.
10. Дати визначення понять «константа» та «змінна».
11. Охарактеризувати розгалужений обчислювальний процес.
12. В якому блоці, шо використовується в розгалуженому обчислювальному процесі, перевіряється логічна умова?
13. Які типи логічних відносин використовуються в блоці умови?
14. Охарактеризувати циклічний обчислювальний процес.
15. Який цикл називається вкладеним?
16. Дати визначення масиву.
17. Які типи масивів Вам відомі, охарактеризувати їх.
18. Сформулювати мету сортування.
19. Перелічити методи сортування.
20. Сформулювати основну вимогу до методів сортування масивів.
21. Скільки порівнянь потребують ефективніші алгоритми сортування?

2 ПОБУДОВА ЛІНІЙНИХ АЛГОРИТМІВ

Лінійним прийнято називати обчислювальний процес, операції якого виконуються послідовно, в порядку їх запису. Кожна операція є самостійною та не залежить від будь-якої вимоги. Блоки, що відображають такі операції, на схемі розташовуються в лінійній послідовності.

Лінійні обчислювальні процеси застосовуються під час обчислення арифметичних виразів.

Приклад розв'язання здачі. Скласти схему алгоритму

$$M = \frac{K + 4F}{c + a}$$
$$c = F + 2a$$
$$a = K + q + b$$
$$b = 21{,}4$$

Вивести на друк *M, a*.

Розв'язання. Умови задачі не передбачають перевірки будь-яких умов або повторення виконання операцій розрахунків, тому обчислюваний процес для розв'язання задачі є лінійним.

Перш ніж будувати блок-схему необхідно дати відповіді на такі питання:

1. Значення яких параметрів надані (тобто є константами в даній задачі)?
Відповідь: $b = 21.4$.
2. Значення яких змінних повинен ввести користувач?
Відповідь: *F, K, q*.
3. Значення яких змінних розраховуються та в якому порядку?
Відповідь: *a, c, M*.
4. Значення яких змінних потрібно вивести на екран?
Відповідь: *a, M*.

Блок-схему алгоритму лінійного обчислюваного процесу, побудовану за вказаними принципами, зображено на рисунку 2.1.

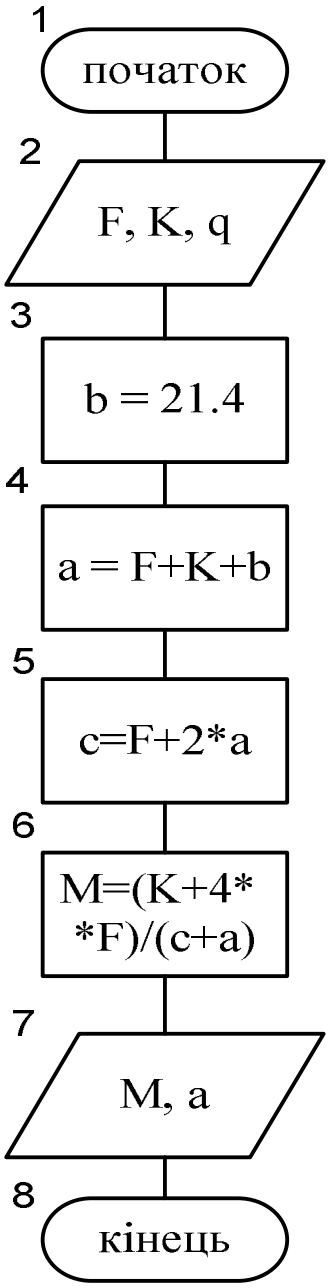


Рисунок 2.1 – Алгоритм лінійного обчислюваного процесу

Блок 1 та 8 зображеної блок-схеми – це блоки початку та кінця алгоритму, блоки 2 та 7 – блоки вводу/виводу даних, блок 3 – блок присвоєння, блоки 4–6 – блоки розрахунків.

У таблиці 2.1 наведені завдання до практичних занять та самостійного вивчення теми «Побудова лінійних алгоритмів».

Таблиця 2.1 – Варіанти індивідуальних завдань для побудови алгоритму лінійного обчислювального процесу

Номер варіанта	Теоретичне питання	Практичне завдання: скласти схему алгоритму
1	2	3
1	Назвати способи опису алгоритмів	$y = \frac{cx^2 + mb^3 + \ln x + 7}{b + m + \sqrt{c}}$ $m = 2x + \sqrt{x}$ $b = ac^2 - m$ Вивести на друк значення змінних y, m, b
2	Надати визначення змінних	$b = a + \sin c$ $a = c + 2$ $s = \sqrt{a + \cos(a + b) - b^2}$ Вивести на друк значення змінних s, a, b
3	Назвати етапи розв'язання задачі	$a = b + c$ $s = (a + b)\sqrt{a^2} + f^3 + b$ $b = c + f^3$ $c = 11$ Вивести на друк значення змінних a, s, b
4	Надати визначення поняття «програма»	$a = y$ $z = \frac{1 - y}{1 + y} 2fy + a^2$ $f = 2y + \sin a$ Вивести на друк значення змінних a, z, f
5	Охарактеризувати типи алгоритмів	$s = (a + b)\sqrt{a^2 + f^2} + b$ $f = 2a + b$ $b = 3.33$ Вивести на друк значення змінних s, f, b
6	У чому полягає сутність властивості алгоритму «масовість»?	$Q = \frac{(p + m)^2}{c + a}$ $p = m^2 + 2 + c_1$ $c_1 = c + a$ де a – довільне значення. Вивести на друк значення змінної Q

Продовження таблиці 2.1

1	2	3
7	У чому полягає сутність властивості алгоритму «дискретність»?	$x = 3y + 2b$ $y = b^2 + 3$ $m = 2 - \frac{3x}{3+b} - \frac{y}{0.9-b}$ $b = 9$ Вивести на друк значення змінних x, y, m
8	Надати визначення поняття «алгоритм»	$y = ax^2 + bx + c$ $c = \sqrt{ax} + d$ $x = d + b$ $a = 0.24,$ де b – довільне число. Вивести на друк значення змінних y, c, x
9	У чому полягає сутність властивості алгоритму «скінченність»?	$p = \frac{a + \ln b + l^x}{d + k}$ $d = a^2 + k1$ $b = 2$ $K = (d + a + b)^3$ Вивести на друк значення змінної p, d, K
10	Які існують типи алгоритмів?	$M = \frac{K^2 + 4 \times F}{c + a}$ $c = F + 2a$ $a = K + q + b$ $b = 21.4$ Вивести на друк значення змінних M, a
11	Назвати способи опису алгоритмів	$D = \frac{B + A^2}{K + P}$ $P = A + B^3$ $K = \ln\lvert A + B^3 \rvert$ $B = -0.1$ Вивести на друк значення змінних K, P, D

Продовження таблиці 2.1

1	2	3
12	Назвати етапи розв'язання задачі	$F = \frac{3 \cdot (m + a) + 2 \cdot x^2 + \sin x}{\ln x}$ $m = \cos x + \sin x$ $x = b$ $b = 4.4$ Вивести на друк значення змінних m, F
13	За допомогою яких операцій присвоюється значення змінній?	$p = \frac{3.2d}{a + 2x} + \frac{mt}{5} + \frac{cx^2}{2m}$ $m = C + 2x^2 + d$ $c = \sqrt{b + 2a}$ Вивести на друк значення змінних p, m, c
14	Надати визначення поняття «константа»	$T = \frac{t_1 + t_2}{a + c^2}$ $t_1 = x + b$ $t_2 = x - b$ $c = a + b$ Вивести на друк значення змінної *T, c,* t_1
15	У чому полягає сутність властивості алгоритму «масовість»?	$I = \frac{j + k^3}{p + n}$ $P = j^3 + n^3$ $K = p^3 + a$ $a = 4.4$ Вивести на друк значення змінної *I, P, K*
16	Охарактеризувати властивості алгоритмів	$V1 = \frac{\nu}{\sqrt{1 - \frac{a^2}{c^2}}}$ $X = V1$ $c = 3 Sin\, x$ Вивести на друк значення змінних *V1, X, c*

Продовження таблиці 2.1

1	2	3
17	Назвати способи опису алгоритмів	$S = \frac{S1 + T + dc}{Sin\,K}$ $K = d + c$ $d = e^x + Sin\,y$ $c = y^2 + x^2$ $z = 2S + y_1$ $y_1 = c + d + k^3$ Вивести на друк значення змінних *S, K, d, c, z, y_1*
18	Надати стислу характеристику лінійного обчислювального процесу	$F = \frac{t^2 + t_1^3 + t_2^4}{\ln x + y^3}$ $X = a + \frac{d}{2}$ $t_1 = (t_2 + t_3)^3$ $t_2 = \frac{z + x}{y}$ $y = 2x + k$ Вивести на друк значення змінних *F, X, t_1, t_2, y*
19	У чому полягає сутність властивості алгоритму «результативність»?	$N = \frac{k + t}{t_2}$ $y = a + b_1$ $b = y^2$ $c = y + b$ $t = \frac{c + b + y}{2k}$ $k = \frac{t_1 + Q}{3}$ Вивести на друк значення змінних *N, y, c, t, k*

Продовження таблиці 2.1

1	2	3
20	Надати визначення поняття «алгоритм»	$A = \frac{f + c^2 + p}{2d}$ $d = q + x$ $t = \frac{a + b + n}{Sin\ p}$ Вивести на друк значення змінних *A, d, t*
21	Охарактеризувати типи алгоритмів	$y = x^2 + k$ $x = z + y_1$ $k = z^2 + y_1^2$ $t = \frac{y}{f + m}$ $m = k + 0.2$ Вивести на друк значення змінних *m, x, y, t, k*
22	Назвати способи опису алгоритмів	$Z = \frac{k^2 + p^3 + n}{Sin\ y + y}$ $y = 0.2x$ $k = y + t^2$ $n = (k + y)^2 + a$ Вивести на друк значення змінних *Z, y, k, n*
23	Назвати етапи підготовки та розв'язання задач	$t = \frac{2s + m^2}{a - x} + \sqrt{s + c^3} - x^2$ $s = \frac{a + 2c}{m} + q$ $m = 2\sqrt{a^2 + c}$ $a = 5{,}6$ Вивести на друк значення змінних *t, s, m*

Продовження таблиці 2.1

1	2	3
24	Дати визначення поняття «лінійний обчислювальний процес»	$Z = \frac{(a+b)(a+c)}{c+d+c_1}$ $d = a + c$ $c_1 = a + b$ $a = 0.24$ $c = -4$ Вивести на друк значення змінних *Z, d, c_1*
25	Надати визначення поняття «алгоритм»	$x = y + k$ $k = \frac{d+c}{m+n}$ $m = c^2 + z$ $z = p_1 + p_2$ $c = 0.13$ $y = 21.2$ Вивести на друк значення змінних *k, m, z*
26	Надати визначення поняття «циклічний алгоритм»	$P = \frac{a+b}{t}$ $t = \frac{x+y}{z}$ $d = P + t + a$ $b = k + m$ $m = -12.1$ Вивести на друк значення змінних *P, d, t*
27	Надати визначення поняття «розгалужений алгоритм»	$a = k + b$ $b = c + d$ $d = t + c$ $m = a + b + \frac{c}{d}$ $t = 2m + d$ $c = 3.3$ Вивести на друк значення змінних *a, b, m, t, d*

Закінчення таблиці 2.1

1	2	3
28	Надати визначення поняття «лінійний алгоритм»	$Z = e^{\frac{x}{t}} + k$ $F = z + d$ $d = x + k$ $k = a + b$ $x = 0.1$ Вивести на друк значення змінних *Z, F, d, k*
29	Надати стислу характеристику лінійного обчислювального процесу	$S = \frac{7}{m + 2t} + \frac{a}{b + cx} + m^2 - 4z$ $z = 3m + at^2$ $m = \sqrt{a + 2cx^2}$ $b = 3.25$ Вивести на друк значення змінних *S, z, m*
30	Охарактеризувати властивості алгоритмів, які Вам відомі?	$V_1 = \frac{v}{v_1 - \frac{a^3}{c^2}}$ $x = vt$ $d = V_1 + a + m + z$ $m = k + v$ $z = x + c$ $c = 3$ Вивести на друк значення змінних *V_1, x, d, m, z*

Контрольні питання

1. Який обчислюваний процес називається лінійним?
2. Які блоки використовують у блок-схемах лінійного обчислюваного процесу?
3. На які питання потрібно дати відповіді, перш ніж почати складати блок-схему алгоритму лінійного обчислюваного процесу?
4. Навести приклад лінійного процесу.
5. Охарактеризувати типи алгоритмів.

3 ПОБУДОВА РОЗГАЛУЖЕНИХ АЛГОРИТМІВ

Обчислювальний процес називається розгалуженим, якщо для його реалізації передбачено кілька напрямів (гілок). Кожен окремий напрям процесу обробки даних є окремою гілкою обчислень.

Обрання напряму залежить від заздалегідь визначеної ознаки, що може стосуватися первинних даних, проміжних або кінцевих результатів. Ознака характеризує властивість даних і має два або декілька значень. Напрям обирається за допомогою перевірки логічного виразу, результатом якого є дві відповіді: «так» або «ні».

Розгалужений процес, що має дві гілки, називається простим, більше за дві гілки – складним. Складний розгалужений процес можна подати за допомогою кількох простих.

Операції, які здійснюються в логічних виразах, є такими:

1) логічне додавання (логічне АБО, OR) – повертає значення ІСТИНА, якщо хоча б один з аргументів має значення ІСТИНА, та значення ХИБНО, якщо всі аргументи мають значення ХИБНО;

2) логічне множення (логічне І, AND) – повертає значення ІСТИНА, якщо всі аргументи мають значення ІСТИНА, та значення ХИБНО, якщо хоча б один з аргументів має значення ХИБНО;

3) логічне НІ (логічне НІ, NOT) – повертає значення ІСТИНА, якщо аргумент має значення ХИБНО та значення ХИБНО, якщо аргумент має значення ІСТИНА.

Існує два типи логічних відносин:

1) відношення еквівалентності. Бінарне відношення називається відношенням еквівалентності, якщо воно рефлексивне, симетричне та транзитивне;

2) відношення порядку. Бінарне відношення називається відношенням порядку, якщо воно рефлексивне та транзитивне.

Приклад розв'язання задачі. Скласти схему алгоритму:

$$y=\begin{cases} Ln(x)+a, & якщо\ \ x>0\ та\ x<1,\\ x^2, & якщо\ 1\le x,\\ x^3, & якщо\ -5<x<0,\\ x, & в\ інших\ \ випадках. \end{cases}$$

$$Z=\begin{cases} 1+y, & якщо\ \ \ y>5,\\ 2\ , & якщо\ \ y=0,\\ y^3, & якщо\ \ \ 0<y\le 5,\\ 4+y^4, & якщо\ \ \ y<0. \end{cases}$$

Вивести на друк *y, Z*.

Розв'язання. У цій задачі для розрахунку *y* та *Z* надано декілька формул. За якою саме з формул знаходити значення цих змінних залежить від того, яка з умов виконується, тому для розв'язання даної задачі потрібно побудувати блок-схему розгалуженого алгоритму.

На початку побудови алгоритму, крім відповідей на питання для лінійного алгоритму, потрібно з'ясувати, між якими з умов стоїть зв'язка логічного додавання (логічне АБО), а між якими – логічного множення (логічне І)?

Якщо між умовами стоїть логічне додавання ($x < -5$ АБО $x > 0$), то результат такого виразу можна подати у наступному вигляді:

Умова 1	Умова 2	Результат
ІСТИНА	ІСТИНА	ІСТИНА
ІСТИНА	ХИБНО	ІСТИНА
ХИБНО	ІСТИНА	ІСТИНА
ХИБНО	ХИБНО	ХИБНО

На блок-схемі цей результат зображується в такий спосіб (рис. 3.1):

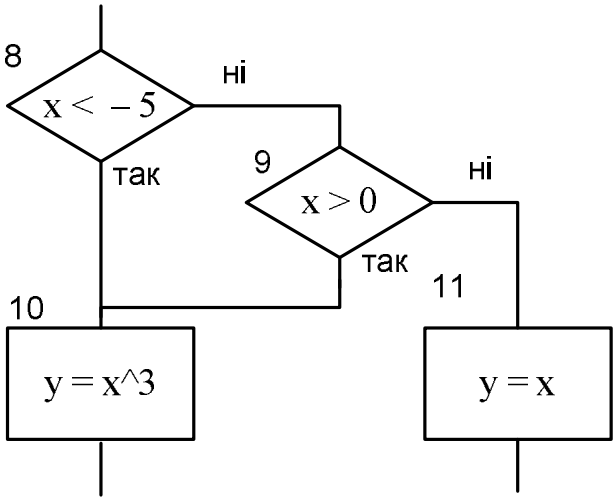


Рисунок 3.1 – Фрагмент алгоритму логічного виразу з оператором логічне АБО

У цьому випадку за виконання умови блоку 8 немає потреби перевіряти другу частину логічного виразу (блок 9), оскільки за виконання хоча б однієї з умов вираз у цілому виконується, а тому можна розраховувати *y* за формулою блоку 10.

У випадку невиконання умови блоку 8 потрібно перевірити виконання другої частини логічного виразу, що записана в блоці 9, якщо вона також не виконується, то *y* розраховується за формулою блоку 11, якщо виконується – за формулою блоку 10.

Якщо між логічними умовами стоїть логічне множення (логічне І) тобто умова записується або так ($x > -5$ *I* $x < 0$), або так ($-5 < x < 0$), то результат такого виразу можна подати у наступному вигляді:

Умова 1	Умова 2	Результат
ІСТИНА	ІСТИНА	ІСТИНА
ІСТИНА	ХИБНО	ХИБНО
ХИБНО	ІСТИНА	ХИБНО
ХИБНО	ХИБНО	ХИБНО

На блок-схемі цей результат зображується в такий спосіб (рис. 3.2):

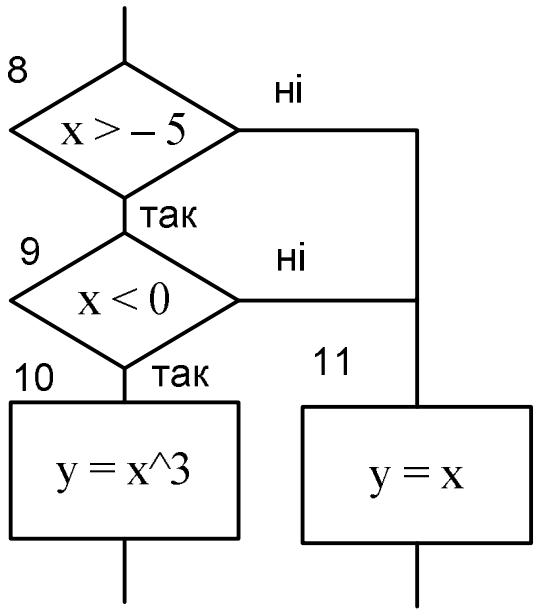


Рисунок 3.2 – Фрагмент алгоритму логічного виразу з оператором логічне І

Оскільки у задачі між умовою $x > -5$ та умовою $x < 0$ стоїть логічне І, а отже, якщо умова блоку 8 не виконується, то немає сенсу перевіряти другу частину виразу, оскільки вираз не буде виконуватись, тому y розраховується за формулою блоку 11. Якщо умова блоку 8 виконується, потрібно перевірити другу частину, яка записана в блоці 9. Невиконання цієї умови також призводить до невиконання всього виразу та розрахунку y за формулою блоку 11. Якщо умова блоку 9 виконується, то вираз в цілому має значення ІСТИНА, а тому y розраховується за формулою блоку10.

Під час побудови блок-схеми алгоритму розгалуженого обчислюваного процесу, крім блоків початку/кінця алгоритму, введення/виведення даних та розрахунків, використовується блок перевірки логічної умови, який має одну вхідну гілку та дві вихідних:

– «так» – гілка, яка відповідає виконанню умови, що міститься в блоці;

– «ні» – гілка, яка відповідає невиконанню умови, що міститься в блоці.

Тому після побудови блок-схеми такого алгоритму доцільно перевірити, чи всі вихідні гілки блоків перевірки логічної умови описані, тобто мають логічне завершення та ведуть до кінця алгоритму.

Після знаходження значення змінної y її потрібно вивести на екран, та обчислити значення змінної Z, перевіряючи відповідні умови, а потім вивести її на екран.

Блок-схему обчислюваного процесу для розв’язання наведеної задачі зображено на рисунку 3.3.

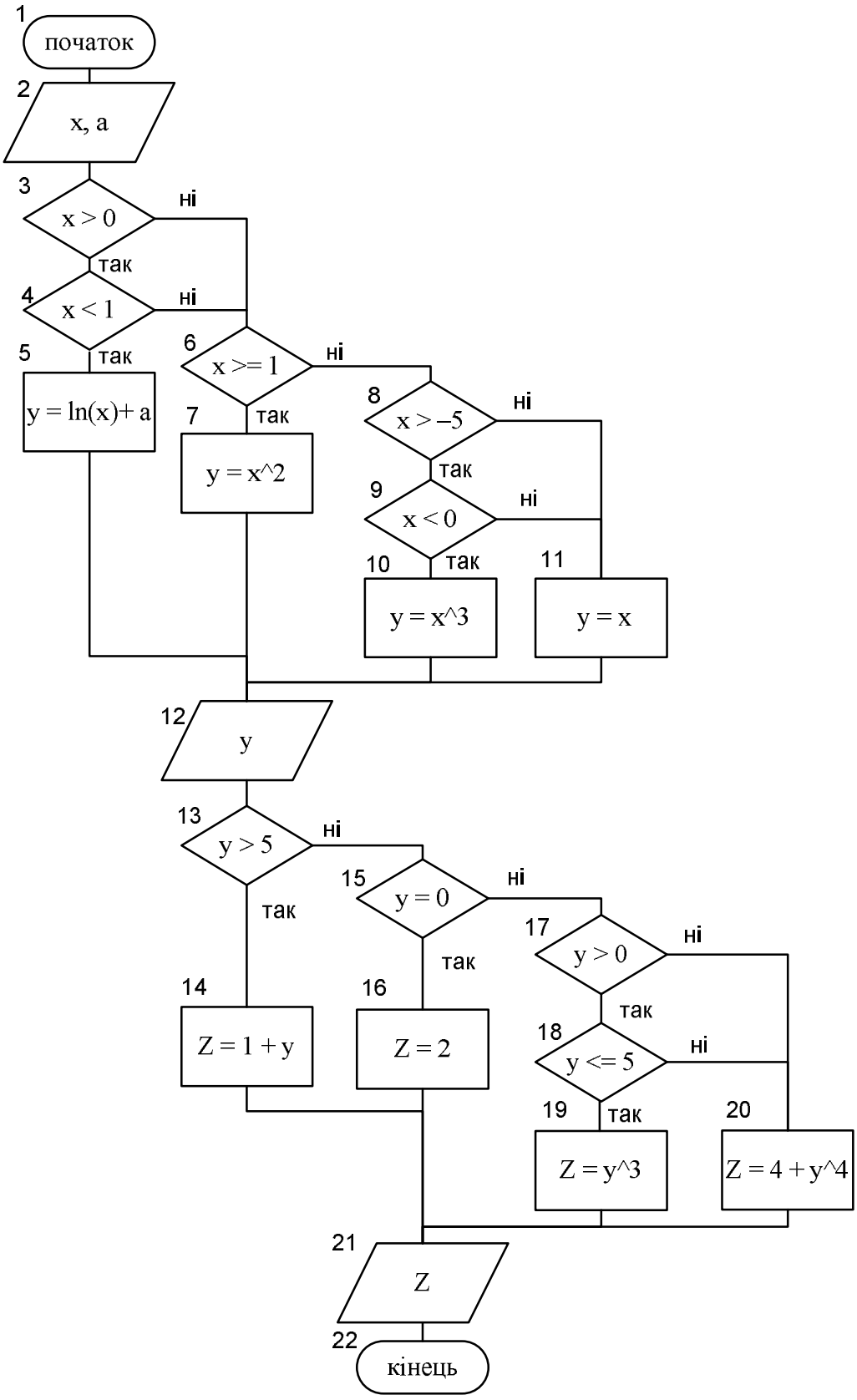


Рисунок 3.3 – Розгалужений алгоритм

Таблиця 3.1 – Варіанти індивідуальних завдань для побудови алгоритму розгалуженого обчислювального процесу

Номер варіанта	Теоретичне питання	Практичне завдання: скласти схему алгоритму
1	2	3
1	Який алгоритм називають розгалуженим?	$Q=\begin{cases}2a-z^x, \text{ якщо } 0<x<5 \text{ та } z\neq 0,\\ \sqrt{x+z+a}, \text{ якщо } 0\le x<9 \text{ та } a=0,\\ a+x+z, \text{ в інших випадках.}\end{cases}$ $S=\begin{cases}2Q, \text{ якщо } Q>15,\\ 3Q, \text{ якщо } Q\le 15.\end{cases}$ Вивести на друк значення змінних *Q, S*
2	Які блоки використовують при побудові розгалуженого алгоритму?	$Z=\begin{cases}ax^3+b\ln 2x^2, \text{ якщо } x>a,\\ \sqrt[6]{\lvert b+cx^2-a\rvert}, \text{ якщо } x<a,\\ \lg(2x+3b^2), \text{ якщо } x=a.\end{cases}$ $Q=\begin{cases}2Z+\ln a, \text{ якщо } Z>153,\\ 2Z+l^x, \text{ якщо } Z\le 153.\end{cases}$ Вивести на друк значення змінних *Z, Q*
3	Що собою являє операція «логічне множення»?	$Y=\begin{cases}x^2+a, \text{ якщо } 0<x<1.5,\\ 2\ln x, \text{ якщо } 1.5<x<3,\\ \sqrt{a+x}, \text{ якщо } 3\le x<5,\\ 12x^2, \text{ в інших випадках.}\end{cases}$ $Z=\begin{cases}3Y, \text{ якщо } Y>3.8,\\ 7Y, \text{ якщо } Y\le 3.8.\end{cases}$ Вивести на друк значення змінних *Y, Z*
4	Назвати типи логічних відносин	$M=\begin{cases}bx^2+\ln 2x+x, \text{ якщо } x\ge 5,\\ \lg\lvert x-a\rvert+b\dfrac{c-x}{a-m}, \text{ якщо } x<5.\end{cases}$ $Q=\begin{cases}e^{x2}+\sqrt{\lvert M-a\rvert}, \text{ якщо } M>56.4,\\ \sin^2 2x+b\sqrt{cx}, \text{ якщо } M<56.4,\\ M+arctg\ x, \text{ якщо } M=56.4.\end{cases}$ Вивести на друк значення змінних *M, Q*

Продовження таблиці 3.1

1	2	3
5	У чому сутність операції «логічного додавання»?	$S = \begin{cases} 2x^m - \ln c^2, \text{ якщо } x^m = 396, \\ x^5 - \lg m, \text{ якщо } x^m = 458, \\ c^5 - \sqrt{x^m}, \text{ якщо } x^m > 458, \\ b + 3mx^m, \text{ якщо } x^m < 458. \end{cases}$ $Q = \ln \frac{S}{m} + x^{m+1},$ $A = \begin{cases} Q^2, \text{ якщо } Q > 0, \\ Q + 1, \text{ якщо } Q \le 0. \end{cases}$ Вивести на друк значення змінних Q, S, X^m, A
6	Які логічні операцій використовуються в логічних виразах?	$T = \begin{cases} a + 2x^в, \text{ якщо } x + c = 2.9, \\ m - 3b^2, \text{ якщо } x + c > 2.9, \\ c + \ln x, \text{ якщо } x + c < 2.9. \end{cases}$ $Q = \frac{X}{T} + \frac{T}{2M} + \sqrt{a + T},$ $S = \begin{cases} a - t, \text{ якщо } T > Q, \\ b + t, \text{ якщо } T \le Q. \end{cases}$ Вивести на друк значення змінних T, Q, S
7	Як позначається в схемах алгоритмів логічне множення?	$T = \begin{cases} x^2 - at, \text{ якщо } c > x > a, \\ cx^5 + 2bx, \text{ якщо } \le a, \\ \sqrt{b + cx^2}, \text{ якщо } x = c, \\ 0, \text{ в інших випадках.} \end{cases}$ $Z = 3.5\ T,$ $Q = \begin{cases} Z + 2T, \text{ якщо } Z \le 29, \\ \ln(Z + T), \text{ якщо } > 29. \end{cases}$ Вивести на друк значення змінних T, Z, Q

Продовження таблиці 3.1

1	2	3
8	Як позначається в схемах алгоритмів операція «логічне додавання»?	$R=\begin{cases} cx^2+\sqrt{b}-c, \text{ якщо } 2<x<5, \\ \sqrt{cx}+a, \text{ якщо } x\le 2, \\ b-2ac, \text{ в інших випадках} \end{cases}$ $Q=3R+2bx$ $T=\begin{cases} 2Q, \text{ якщо } Q<128, \\ Q-3A, \text{ якщо } \ge 128. \end{cases}$ Вивести на друк значення змінних *R, Q, T*
9	Що собою являє операція «кон'юнкція»?	$R=\begin{cases} a+2x, \text{ якщо } x<0 \text{ та } a>2, \\ b-cx, \text{ якщо } x>0 \text{ або } a=0, \\ a^2-2\sqrt{x}, \text{ якщо } x=4 \text{ та } c=7, \\ \sqrt{b+x}, \text{ в інших випадках.} \end{cases}$ $S=\begin{cases} 2R, \text{ якщо } R<12, \\ R-C, \text{ якщо } R\ge 12. \end{cases}$ Вивести на друк значення змінних *R, S*
10	Що собою являє операція «диз'юнкція»?	$R=\begin{cases} cx^2+\lg x, \text{ якщо } 2<x<4, \\ b-cx, \text{ якщо } 4\le x<6, \\ a^3+\lg x, \text{ якщо } 6\le x<8, \\ cx+b^x, \text{ якщо } x\le 2 \text{ або } x\ge 8. \end{cases}$ $S=\begin{cases} 2R, \text{ якщо } R=14.2, \\ 3R, \text{ якщо } R\ne 14.2. \end{cases}$ Вивести на друк значення змінних *R, S*

Продовження таблиці 3.1

1	2	3
11	Який алгоритм називають розгалуженим?	$Y = \begin{cases} x + 2a^m, \ якщо \ \ a + m = 4.6, \\ \sqrt{x + 3a}, \ якщо \ \ a + m \geq 5.2, \\ \ln(a + m), \ в \ інших \ \ випадках \ \ . \end{cases}$ $Z = \begin{cases} 2Y, \ якщо \ \ Y < 16.5, \\ 5Y - C, \ якщо \ \ Y \geq 16.5. \end{cases}$ $Q = a + 2Y - 3Z.$ Вивести на друк значення змінних *Y, Z, Q*
12	Назвати типи логічних відношень	$Y = \begin{cases} 3x, \ якщо \ 2 < x < 4, \\ 5x, \ якщо \ 4 \leq x < 9, \\ 7x, \ якщо \ 9 \leq x \ або \ x \leq 2, \end{cases}$ $Z = 3Y + 2ax - \sqrt[q]{Y},$ $R = \begin{cases} Z + Y, \ якщо \ Z \geq 25, \\ Y - Z, \ якщо < 25. \end{cases}$ Вивести на друк значення змінних *Y, Z, R*
13	Назвати типи логічних відношень	$Y = \begin{cases} \lvert x \rvert + x^2, \ якщо \ \ x < 0, \\ kx + b, \ якщо \ \ x = 0, \ k = 5.35, \ b = 10.11, \\ \ln^2 x, \ якщо \ \ \ x > 0, \end{cases}$ $S = \begin{cases} Y^2, \ якщо \ \ Y \leq 0, \\ \sqrt{Y}, \ якщо \ \ Y > 0. \end{cases}$ Вивести на друк *Y, S*
14	Охарактеризувати властивість алгоритму «детермінованість»	$Z = \begin{cases} b + am^x, \ якщо x = 125, \\ c - b + x, \ якщо \ x = 200, \\ e^x + a, \ якщо \ 0 < x < 125, \\ 10, \ в \ інших випадках. \end{cases}$ $Q = \frac{a}{Z} + \frac{b}{2Z} - c,$ $A = \begin{cases} Q + 3, \ якщо \ Q > 0, \\ Q - 4, \ якщо \ Q \leq 0. \end{cases}$ Вивести на друк значення змінних *Z, Q, A*

Продовження таблиці 3.1

1	2	3
15	Охарактеризувати властивість алгоритму «дискретність»	$Z=\begin{cases} ax^3+b\ln 2x^2, & якщо\ x>a, \\ c\sqrt{\lvert b+cx^2-a\rvert}, & якщо\ x=a, \\ \lg(2x+3b^2), & якщо\ x=a. \end{cases}$ $Q=\begin{cases} 2Z^2+\ln a, & якщо\ Z>153, \\ 2Z^2+e^2, & якщо\ Z\le 153. \end{cases}$ Вивести на друк значення змінних *Z, Q*
16	У чому полягає сутність операції «диз'юнкція»?	$Y=\begin{cases} x\dfrac{b+a}{\sin x}, & якщо\ 0<x<0.5\ та\ a=3.3\ та\ b=1.17, \\ b+\dfrac{x^2+1}{5}, & якщо\ x=0\ або\ 0<b<5, \\ \dfrac{x-\dfrac{x+1}{2}}{2x}, & в\ інших\ випадках. \end{cases}$ $Q=\begin{cases} Y-a, & якщо\ Y\ge 10, \\ Y+b, & якщо\ Y<10. \end{cases}$ Вивести на друк *Y, Q*
17	Охарактеризувати властивість алгоритму «масовість»	$Z=\begin{cases} \sqrt{b+\ln x-2a}, & якщо\ bx=6\ та\ a<2, \\ \sqrt[q]{cx+5a-c}, & якщо\ 6<bx<9\ та\ a>2, \\ \ln\lvert b+ax^2\rvert-b, & в\ інших\ випадках. \end{cases}$ $R=\begin{cases} 2Z, & якщо\ Z\ge 12, \\ 3Z, & якщо\ Z<12. \end{cases}$ Вивести на друк значення змінних *Z, R*
18	Який алгоритм називається розгалуженим?	$Y=\begin{cases} x^{e^{x+a}}+e^{2+a}, & якщо\ 1\le x<2\ та\ a>0, \\ \sin(x+b)+\dfrac{x}{9}, & якщо\ -1\le x<1\ або\ 0<b<3, \\ \ln\lvert b+ax^2\rvert-b, & якщо\ -5\le x<1. \end{cases}$ $P=\begin{cases} Y+b^2, & якщо\ Y\le 0, \\ Y+a^2, & якщо\ Y>0. \end{cases}$ Вивести на друк значення змінних *Y, P*

Продовження таблиці 3.1

1	2	3
19	Який алгоритм називають циклічним?	$Y=\begin{cases}\ln x^3+\sqrt{x+c}, & якщо\ 3<x<6\ та\ c>0,\\ \dfrac{x^2}{a+x}+\sqrt{ax}, & якщо\ x>10\ або\ 0<a<6,\\ x\ln\lvert x\rvert, & в\ інших\ випадках.\end{cases}$ $Z=\begin{cases}\sqrt{Y}, & якщо\ Y>0,\\ \sqrt[3]{Y}, & якщо\ Y\le 0.\end{cases}$ Вивести на друк значення змінних *Y, Z*.
20	У чому полягає сутність операції логічного додавання?	$Y=\begin{cases}1+2\sin^2(x+0.1a), & якщо\ -1<x<1\ або\ 1<a<2,\\ x^2+\sqrt{\lvert x+a+b\rvert}, & якщо\ -1\le x\ та\ a<0\ та\ b<0,\\ tg\,x, & в\ інших\ випадках.\end{cases}$ $Z=\begin{cases}2a+Y, & якщо\ Y>0\ та\ a>0,\\ 2b-Y, & якщо\ Y<0\ або\ b<0.\end{cases}$ Вивести на друк значення змінних *Y, Z*
21	Який алгоритм називають розгалуженим?	$Y=\begin{cases}\cos^2\lvert x+a\rvert, & якщо\ -2<x<-1\ та\ -3<a<-4,\\ \sqrt[3]{x+\sin^2 x+b} & якщо\ x\ge -1\ або\ -2<b<-3,\\ ctgx, & якщо\ x=-5.\end{cases}$ $P=\begin{cases}\ln Y, & якщо\ Y>0,\\ \ln^2\lvert Y\rvert, & якщо\ Y\le 0.\end{cases}$ Вивести на друк значення змінних *Y, P*
22	Навести способи опису алгоритмів	$Y=\begin{cases}\sqrt[3]{x+e^{x+a+b}}, & якщо\ 7<x<9\ або\ a<1\ та\ b>3,\\ tg\sqrt{x}, & якщо\ x=3,\\ \ln\lvert x-a-b\rvert, & якщо\ x=1\ та\ a=2\ або\ b=-1.\end{cases}$ $T=\begin{cases}2Y, & якщо\ Y>5,\\ 3\lvert Y\rvert, & якщо\ Y\le -5.\end{cases}$ Вивести на друк значення змінних *Y, T*

Продовження таблиці 3.1

1	2	3
23	Охарактеризувати властивість алгоритму «дискретність»	$Y=\begin{cases}\dfrac{a+\sqrt{x}}{\sqrt{100-b}}, \text{ якщо } 1<x<2 \text{ та } 4<a<7 \text{ або } b=3,\\ \lvert x\rvert, \text{ якщо } x<-1,\\ \ln x^3, \text{ якщо } x=0.75.\end{cases}$ $P=\begin{cases}Y+\sqrt{\lvert a\rvert}, \text{ якщо } Y>0 \text{ або } a<0,\\ Y, \text{ якщо } Y=0.\end{cases}$ Вивести на друк значення змінних *Y, P*
24	Охарактеризувати типи алгоритмів	$Y=\begin{cases}\dfrac{2^{-x}\sqrt{x+\lvert y_1\rvert}}{2}, \text{ якщо } 1<x<10,\\ \sin z+\sqrt{e^x}, \text{ якщо } 0<x\le 1,\\ x^2+4, \text{ в інших випадках}.\end{cases}$ $S=\begin{cases}2Y+\ln a, \text{ якщо } Y>2.2,\\ 2Y^2, \text{ якщо } Y\le 2.2.\end{cases}$ Вивести на друк значення змінних *Y, S*
25	Охарактеризувати циклічний алгоритм	$Q=\begin{cases}ctg^2 x, \text{ якщо } 0\le x<1,\\ tg\, x, \text{ якщо } 1\le x<2 \text{ або } x=3,\\ \ln x, \text{ якщо } 4<x<5 \text{ та } 9<x<10.\end{cases}$ $T=QZ,$ $A=\begin{cases}T^2, \text{ якщо } T>0,\\ \lvert T+1\rvert, \text{ якщо } T\le 0.\end{cases}$ Вивести на друк значення змінних *Q, T, A*

Продовження таблиці 3.1

1	2	3
26	Описати типи структур алгоритмів	$F = \begin{cases} e^x + a, & якщо \quad -3 \le x < 3, \\ \ln x^2 + b, & якщо \quad 5 \le x < 7, \\ \lvert x \rvert, & якщо \quad x = -10 \quad або \quad -20 < x < -11. \end{cases}$ $P = F + T$ $Z = \begin{cases} \cos P, & якщо \quad P < 0, \\ \ln(P), & якщо \quad P > 0, \\ 5, & якщо \quad P = 0. \end{cases}$ Вивести на друк значення змінних *F, P, Z*
27	Описати способи опису алгоритмів	$Q = \begin{cases} zx + ax, & якщо\ -1 < x < 3\ та\ 0 < z < 1, \\ \dfrac{z}{4x} + ax, & якщо\ 3 < x < 12, \\ a + z^2, & в\ інших\ випадках. \end{cases}$ $S = \begin{cases} Q^3, & якщо\ Q > 17, \\ Q^2, & якщо\ Q \le 17. \end{cases}$ Вивести на друк значення змінних *Q, S*
28	Охарактеризувати типи алгоритмів	$T = \begin{cases} \ln x + \ln b + \sqrt{a}, & якщо\ 0 < x < 1\ та\ 2 < b < 3\ та\ 4 < a < 5, \\ \sin x + \cos\dfrac{b}{a}, & якщо\ -1 < x \le 0\ або\ 0.5 < b \le 2\ та\ a \ne 0, \\ a + b + x, & в\ інших\ випадках \end{cases}$ $D = \begin{cases} 2T, & якщо\ T \ge 10, \\ 3\sqrt{\lvert T \rvert}, & якщо\ T < 10. \end{cases}$ Вивести на друк значення змінних T, D
29	Дати визначення циклічного алгоритму	$Q = \begin{cases} 10a - zx, & якщо\ 0 < x < 1\ та\ 0 < z < 1, \\ x^2 + \sqrt{x}, & якщо\ -2 < x < 1\ або\ 3 < x < 4, \\ a + x - z^2, & якщо\ -4 < z < 3\ та\ x > 10. \end{cases}$ $T = \begin{cases} Q^2, & якщо\ Q > 0, \\ cos(Q), & якщо\ Q = 0, \\ \lvert Q \rvert, & якщо\ Q < 0. \end{cases}$ Вивести на друк значення змінних *Q, T*

Закінчення таблиці 3.1

1	2	3
30	Описати типи структур алгоритмів	$R=\begin{cases}\sqrt{bx+c^2}, & якщо\ 1<x<2\ \ та\ \ 1<b<2,\\ \sqrt{b+xb}, & якщо\ -1<x<0\ \ та\ \ 3<b<4,\\ c\lvert x\rvert+b, & якщо\ x<-10\ \ та\ \ 7<b<8.\end{cases}$ $Q=\frac{r}{5}+10bx$ $A=\begin{cases}10Q, & якщо\ Q>15,\\ 20Q, & якщо\ Q\le 15.\end{cases}$ Вивести на друк значення змінних *R, Q, A*

Контрольні питання

1. Який обчислюваний процес називається розгалуженим?
2. Від чого залежить обрання напрямку обчислення в розгалуженому обчислювальному процесі?
3. Що таке логічна умова?
4. Який розгалужений процес називається простим?
5. Який розгалужений процес називається складним?
6. Як операції в логічних виразах Вам відомі? Що є результатом кожної з них?
7. Які типи логічних відносин Вам відомі?
8. Дати визначення відношення еквівалентності.
9. Дати визначення відношення порядку.
10. Які блоки використовують у блок-схемах розгалуженого обчислюваного процесу?
11. На які питання потрібно дати відповіді, перш ніж складати блок-схему алгоритму розгалуженого обчислюваного процесу?
12. У чому полягає особливість блоку перевірки логічної умови?
13. У чому полягає різниця між блок-схемою для алгоритмізації логічного додання та логічного множення?
14. Навести результати логічного додавання у табличному вигляді.
15. Навести результати логічного множення у табличному вигляді.
16. Які особливості побудови блок-схеми алгоритму розгалуженого обчислювального процесу?
17. Навести приклад розгалуженого процесу.

4 ПОБУДОВА ЦИКЛІЧНИХ АЛГОРИТМІВ

Цикл – це фрагмент обчислювального процесу, що повторюється багаторазово.

Обчислювальні процеси, в яких виконуються одні й ті самі дії з різними даними, називаються циклічними.

Послідовність операторів, що повторюється, називається тіло циклу.

Лічильник циклу – це змінна, значення якої змінюється після кожного повторення циклу, та яке визначає закінчення повторів циклу.

Параметр циклу – це будь-який арифметичний вираз, що не містить керувальної змінної (лічильника циклу).

Для організації циклу потрібно передбачити такі значення:

1) початкове значення змінної, яке буде змінюватися під час повторення циклу;

2) кінцеве значення цієї змінної;

3) крок зміни перед кожним новим повторенням циклу. Необхідно контролювати поточне значення лічильника циклу для перевірки умови виходу з циклу. Такою умовою може бути:

а) перевищення лічильником циклу кінцевого значення;

б) виконання заданої кількості повторень;

в) досягнення заданої точності розрахунків.

Розрізняють арифметичні та ітераційні цикли:

– в арифметичних циклах кількість повторень є знаною заздалегідь або може бути визначена на підставі закону зміни лічильника циклу;

– в ітераційних циклах кількість повторень тіла циклу заздалегідь не є відомою, а цикл повторюється доти, доки не буде виконана умова виходу з циклу [6].

Алгоритм, тіло циклу якого містить ще один цикл, називається алгоритмом із вкладеним циклом.

Цикл, в тіло якого вкладені команди іншого циклу, називається зовнішнім циклом, а цикл, який вкладено – вкладеним або внутрішнім циклом.

Приклади розв’язання задач.

Задача 1. Скласти схему алгоритму циклічного обчислюваного процесу для $x \in [1;5],\ \Delta x = 0{,}1$.

$$y = x\left(\frac{\sum_{i=1}^{10} i^2}{k!} + \prod_{j=1}^{k} j^3\right).$$

Вивести до друку значення змінних *x, y*.

Розв'язання. Циклічному процесу властиве повторення набору операцій над різними даними. У наведеній задачі існує чотири таких набори:

1) підрахунок суми $\sum_{i=1}^{10} i^2$;

2) підрахунок факторіалу числа *k!*;

3) підрахунок добутку $\prod_{j=1}^{k} j^3$;

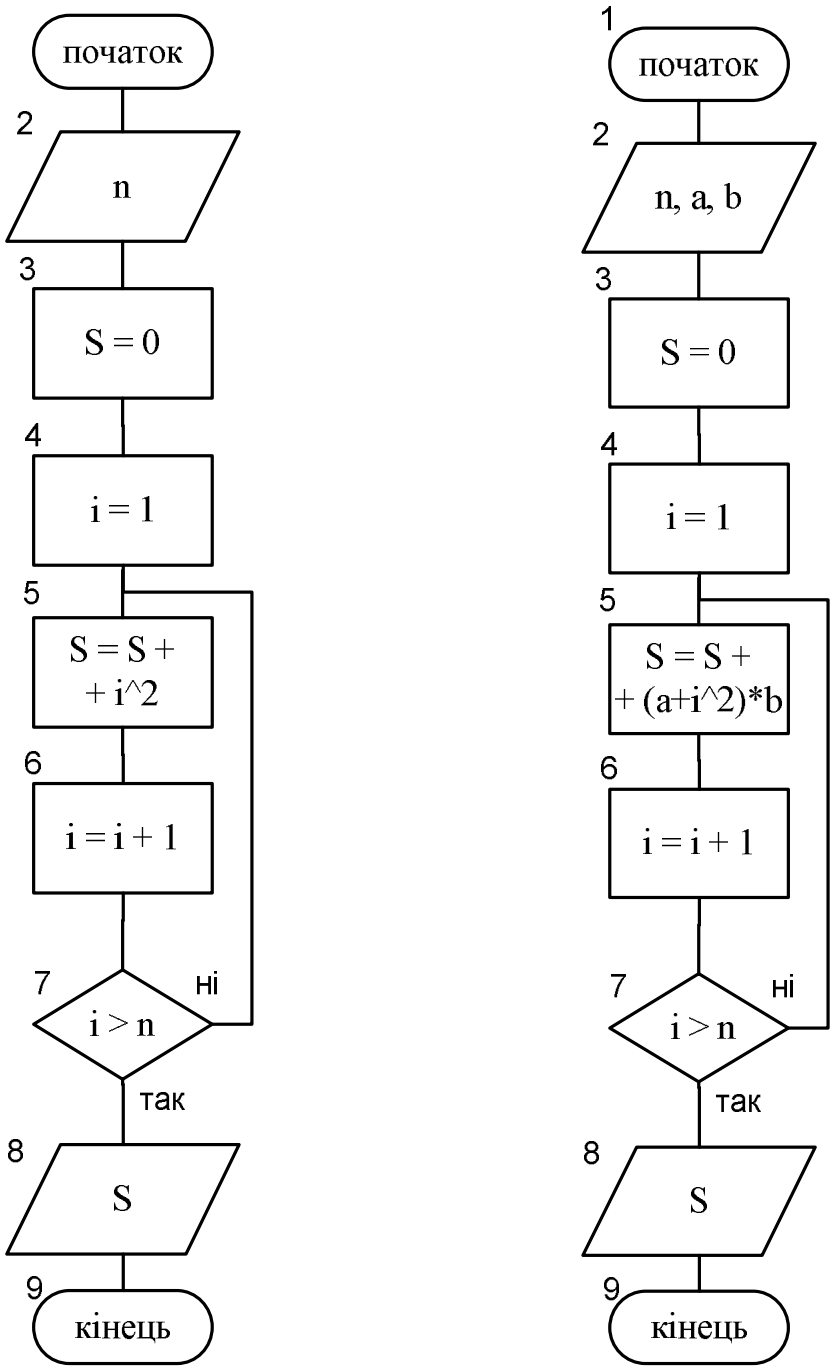
4) підрахунок значення *y* для кожного зі значень параметра *x*, який змінюється на інтервалі від 1 до 5 з кроком 0,1.

Перш ніж будувати блок-схему алгоритму для розв'язання всієї задачі розглянемо кожен з чотирьох пунктів.

Підрахунок суми. Числа знизу та зверху значка суми позначають границі зміни значення змінної *i*. Позначимо змінну, в якій буде зберігатися результат обчислень через *S*, тоді $S = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 + 7^2 + 8^2 + 9^2 + 10^2$. У цьому випадку виконуються операції піднесення до другого ступеня значення змінної *i* та додавання до змінної, значення якої змінюється від 1 до 10 з кроком 1. Якщо розбити цей процес на окремі кроки, то кожен крок буде полягати в додаванні до попереднього результату значення змінної *i*, піднесеного до другого ступеня та зміни цього значення на одиницю. У математичних виразах такі дії можна записати у вигляді: $S = S + i^2$, $i = i + 1$. Блок-схема алгоритму розрахунку в даному випадку матиме вигляд, наведений на рисунку 4.1,а, при цьому параметр *n* задано рівним 10.

З точки зору реалізації на комп'ютері формула $S = S + i^2$ має свою специфіку: на першому кроці значення змінної *S* є невідомим; у клітині пам'яті, яку буде відведено під цю змінну, може міститися будь-яке значення, наприклад, код символу @), що може спотворити результат підрахунків суми. Тому, перш ніж починати цикл, потрібно надати змінній *S* значення, яке не змінить результат, для суми таким значенням є нуль.

У випадку складнішого виразу під знаком суми блок-схема не змінюється, іншою стає формула в блоці 5 та кількість змінних, значення яких уводяться в другому блоці. На рисунку 4.1,б зображено блок-схему для розрахунку суми: $\sum_{i=1}^{10}(a + i^2)b$.



а б

Рисунок 4.1 – Блок-схема алгоритму підрахунку суми

У наведених циклах лічильником є змінна *i*, початкове значення якої дорівнює $n = 1$ (блок 4), кінцеве значення – $n = 10$, крок зміни лічильника є рівним 1 (блок 6), а умовою виходу з циклу є перевищення лічильником кінцевого значення ($i > n$), перевірка якої виконується у блоці 7. Доти, доки поки умова блоку 7 не виконується, процес повертається до блоку 5, де виконується тіло циклу (формула блоку 5) та збільшення значення лічильника циклу на крок (блок 6).

Підрахунок факторіалу. Факторіалом числа k ($k!$) є добуток усіх натуральних чисел від 1 до k включно. Позначимо змінну, яка містить значення факторіалу, через F, тоді $F = 1 \cdot 2 \cdot 3 \cdot ... (\mathrm{k} - 1) \cdot \mathrm{k}.$

Якщо розбити процес підрахунків на окремі кроки, то кожен крок складатиметься з множення результату, отриманого на попередньому кроці на нове значення, тобто $F = \mathrm{F} \cdot \mathrm{i}.$, до того ж значення i змінюється від 1 до k з кроком, рівним 1.

Блок-схему алгоритму розрахунку факторіалу зображено на рисунку 4.2.

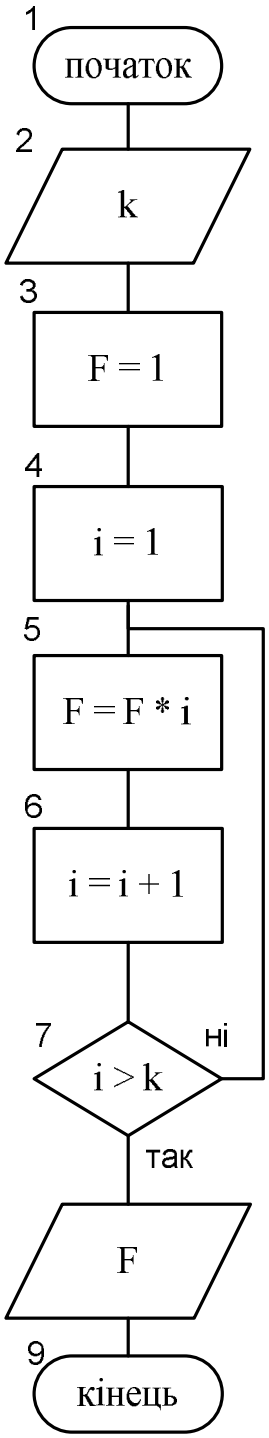


Рисунок 4.2 – Блок-схема алгоритму розрахунку факторіалу $k!$

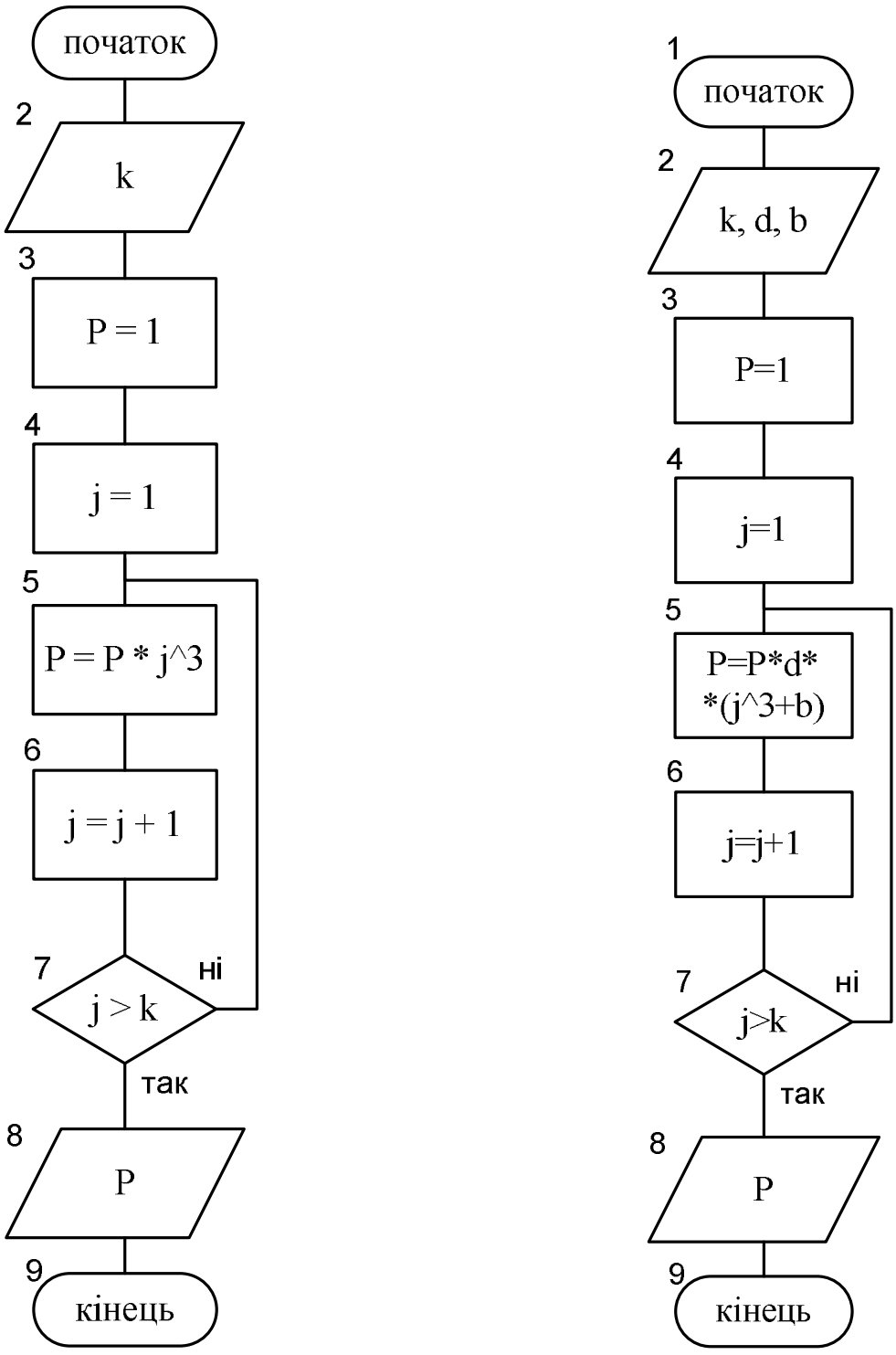
Так само, як і в підрахунку суми перш ніж виконувати цикл потрібно змінній, яка буде містити результат, надати значення, яке не змінить результат. Для операції добутку це значення дорівнює 1 (блок 3).

У наведеному циклі лічильником є змінна *i*, початковим значенням якої є 1 (блок 4), кінцевим – *k*, кроком зміни лічильника – 1 (блок 6), а умовою виходу з циклу – перевищення лічильником кінцевого значення ($i > k$), перевірка якого виконується в блоці 7. Поки умова блоку 7 не виконується, процес повертається до блоку 5, де виконується тіло циклу (формула блоку 5) та збільшення значення лічильника циклу на крок (блок 6).

Підрахунок добутку. Побудова блок-схеми алгоритму добутку є цілком аналогічною до побудови алгоритму підрахунку суми, за виключенням того, що змінній-результату (в цьому випадку *P*) слід надати попереднє значення, яке дорівнює 1, а в тілі циклу застосувати операцію множення попереднього значення змінної лічильника на наступне (рис. 4.3,а).

Якщо під знаком добутку стоїть складніша формула, наприклад: $\prod_{j=1}^{k} d(j^3 + b)$, то блок-схема алгоритму матиме вигляд, наведений на рисунку 4.3,б.

У блок-схемах алгоритмів циклічних обчислюваних процесів для зручності прийнято використовувати блок циклу, до якого заносять початкове, кінцеве значення лічильника циклу та крок зміни лічильника (якщо крок дорівнює 1, його не вказують). Такий блок замінює блоки 4, 6 та 7 і має два вхідні потоки (від попереднього блоку та від кінця тіла циклу в разі невиконання умови виходу з циклу) та два вихідні потоки (до блоку тіла циклу та до наступних блоків алгоритму в разі виконання умови виходу з циклу).



а б

Рисунок 4.3 – Блок-схеми для розрахунку добутків

Блок-схеми розрахунків суми, факторіалу та добутку з блоком циклу наведено на рисунках 4.4,а, 4.4,б та 4.4,в відповідно.

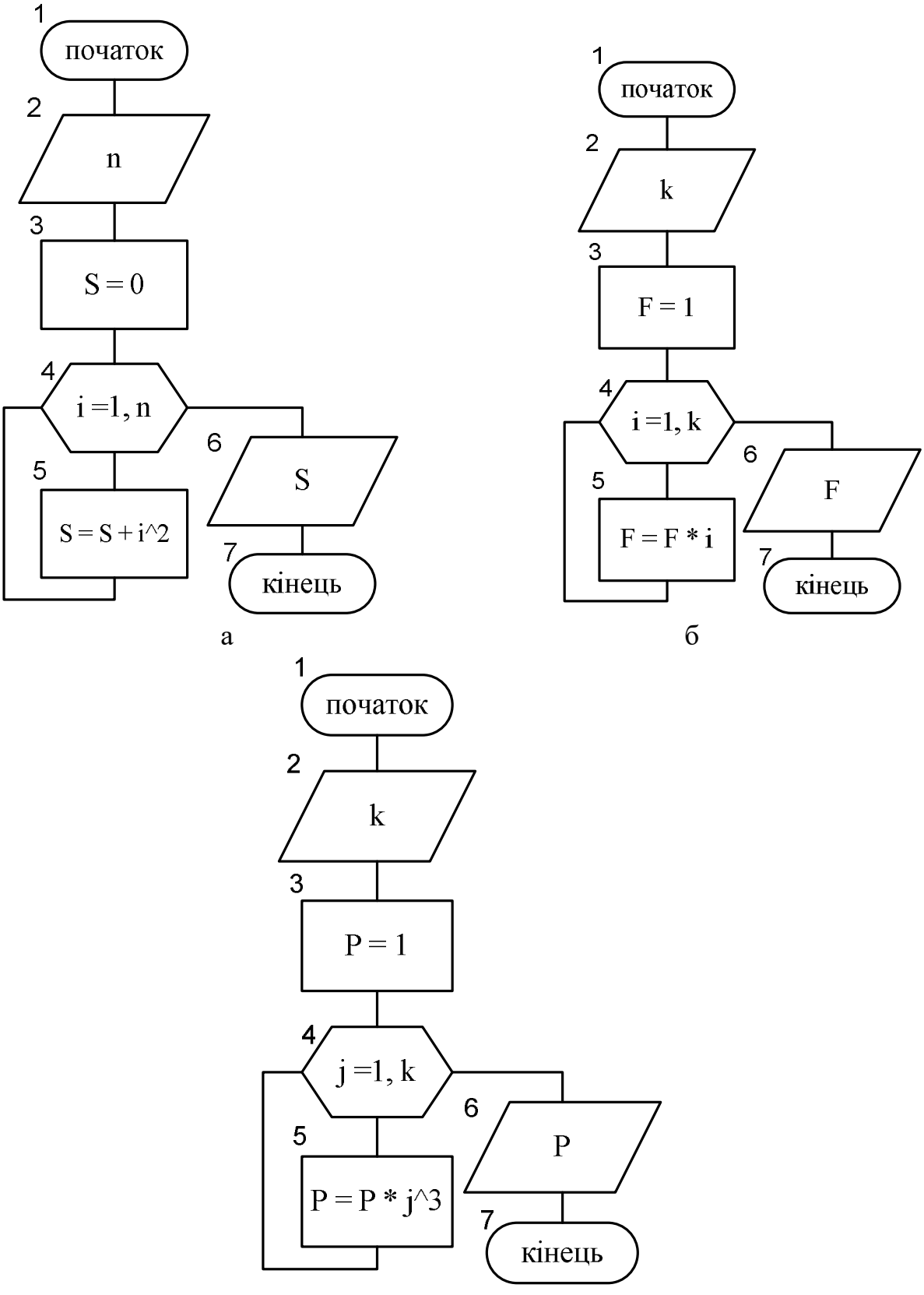


Рисунок 4.4 – Блок-схеми з блоком циклу

Кінцевий вигляд схеми алгоритму для всієї задачі наведено на рисунку 4.5.

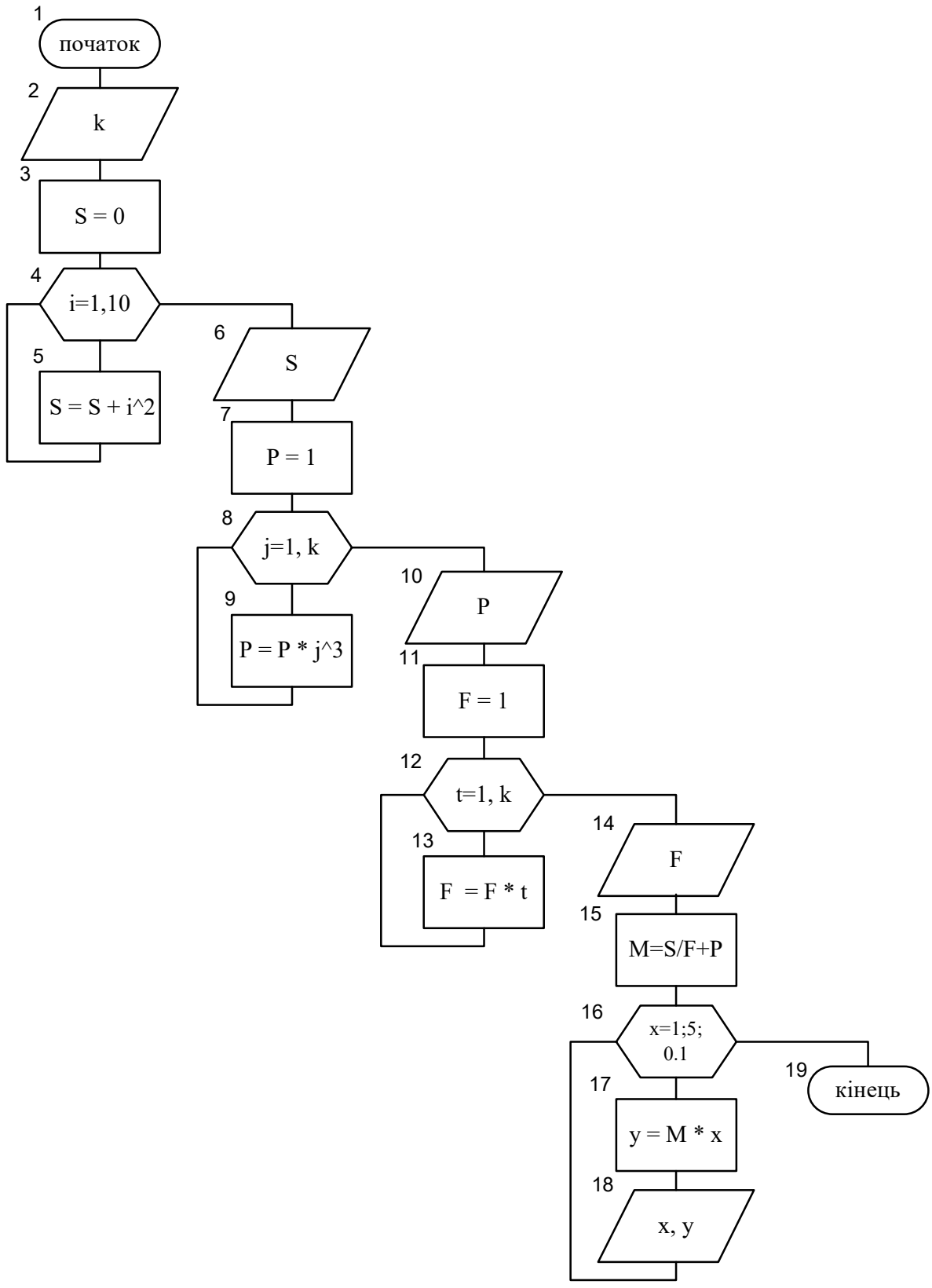


Рисунок 4.5 – Блок-схеми алгоритму циклічного обчислювального процесу

Підрахунок *y* (рис. 4.5). Значення *y* залежить від значення *x*, яке набуває не одне, а декілька значень (*x* = 1, 1.1, 1.2, …, 4.8, 5), тому для знаходження та виведення значення *y* слід організувати цикл відносно змінної *x*.

У цьому циклі лічильником є змінна x, початковим значенням якої є значення 1, кінцевим – значення 5, кроком зміни – значення 0.1, а умовою виходу з циклу – перевищення лічильником циклу кінцевого значення (блок 16). Тіло циклу складають блоки 17 (підрахунок значення *y*) та 18 (виведення значень *x* та *y* на екран).

Оскільки значення факторіалу, суми та добутку не залежать від змінної *x*, то вони підраховуються окремо до початку організації циклу відносно змінної *x*.

Задача 2. Скласти схему алгоритму циклічного обчислюваного процесу із вкладеним циклом:

$$y = \sum_{x=1}^{K} \left(\frac{1}{x^2} + \frac{\prod_{n=1}^{m} \frac{1}{n^2 x}}{x!} \right).$$

Вивести до друку значення змінної *y*.

Розв'язання. Початковим блоком блок-схеми є блок введення, в якому вводяться значення *K* та *m*.

У цій задачі потрібно організувати зовнішній цикл відносно змінної *x*, тіло якого будуть становити:

– блок суми значень *y*;

– блок вкладеного циклу для підрахунку факторіалу *x*;

– блок вкладеного циклу для підрахунку добутку, який зі свого боку залежить від значення *x*.

Але перш ніж організовувати зовнішній цикл змінної *y*, їй потрібно присвоїти значення, яке не змінить результат додавання, оскільки в ній поступово буде накопичуватися значення суми.

Блок-схему алгоритму циклічного обчислювального процесу із вкладеними циклами наведено на рисунку 4.6.

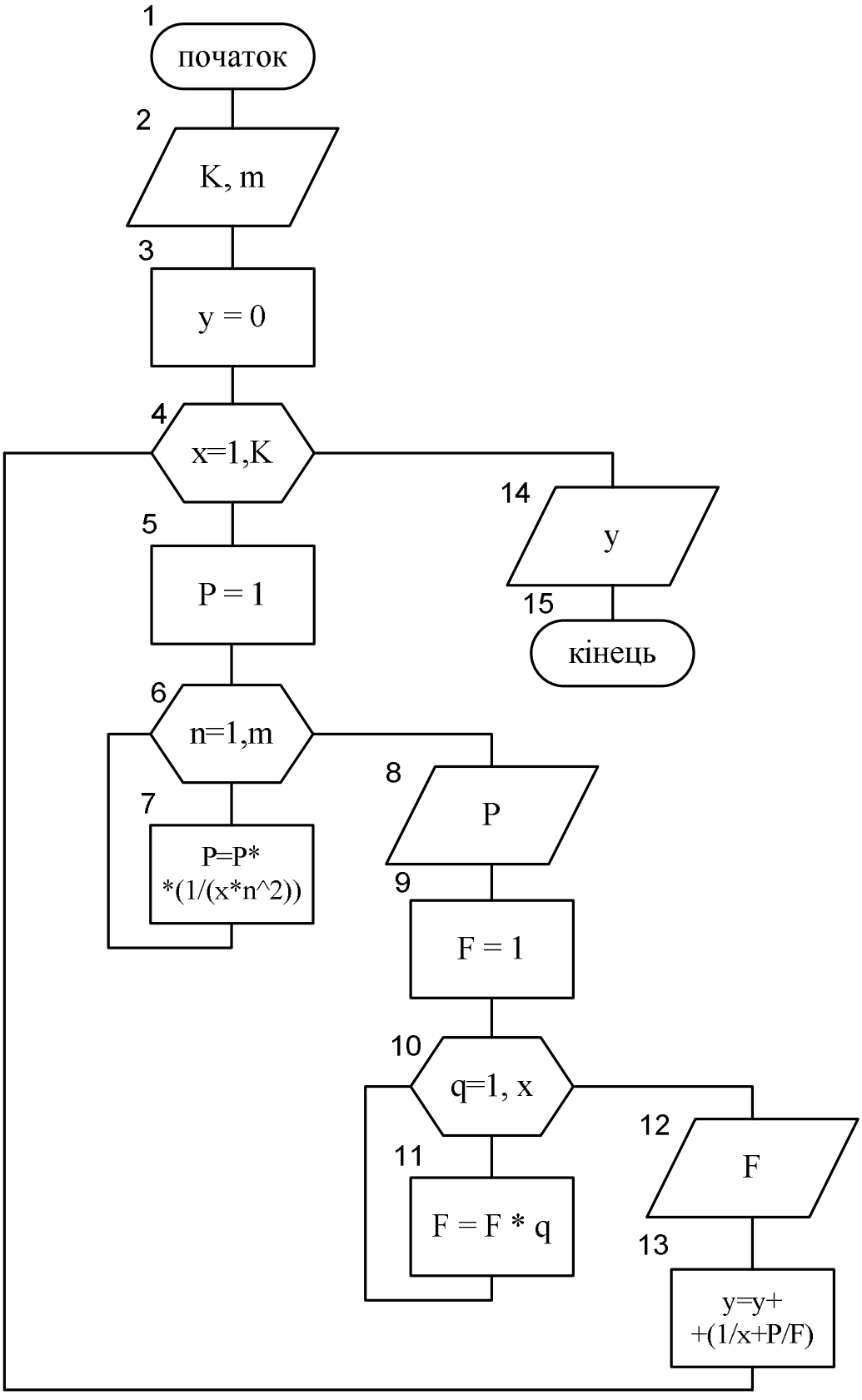


Рисунок 4.6 – Блок-схема алгоритму із вкладеними циклами

Задача 3. Скласти схему алгоритму циклічного обчислюваного процесу із вкладеними циклами:

$$y = \sum_{x=0}^{m} \left(\frac{1}{\sum_{i=1}^{n} \frac{1}{x^2} + 1} + \prod_{i=1}^{n} \frac{x}{i!} \right).$$

Вивести до друку значення змінної *y*.

Розв'язання. У цьому випадку задача ускладнена тим, що до циклу відносно змінної *x* вкладено цикл підрахунку суми та цикл підрахунку добутку, до якого зі свого боку вкладено цикл підрахунку факторіалу значення лічильника циклу. Отже, маємо подвійне вкладення.

У цій задачі необхідно:

– відкрити зовнішній цикл по змінній *x* $\sum_{x=0}^{m}( );$

– обрахувати в циклі значення суми $\sum_{i=1}^{n} \frac{1}{x^2} + 1;$

– відкрити ще один вкладений цикл по змінній *i*, в якому обрахувати факторіал *i!* та добуток $\prod_{i=1}^{n} \frac{x}{i!}$.

Блок-схема алгоритму циклічного обчислювального процесу із подвійним вкладенням циклу наведено на рисунку 4.7.

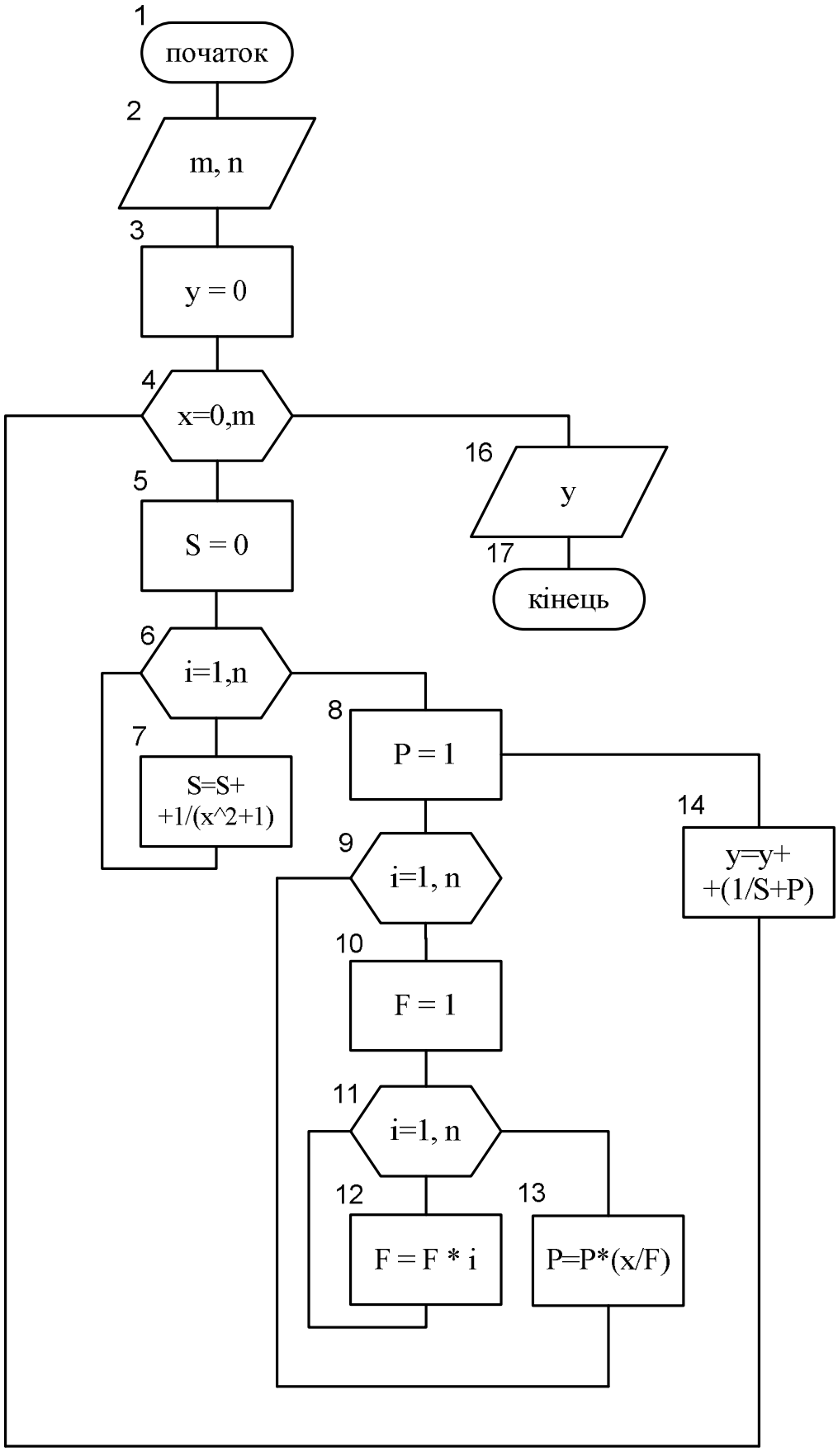


Рисунок 4.7 – Блок-схема алгоритму з подвійним вкладенням циклів

Задача 4. Скласти схему алгоритму обчислюваного процесу:

$$D = \begin{cases} x^2 + abx, & якщо \quad 0 \le x \le 2, \\ |x| - a - b, & якщо \quad 2 < x \le 3 \quad або \quad -3 \le x \le -2, \\ abx^2, & в \quad інших \quad випадках. \end{cases}$$

$$C = D + x$$

у випадку, якщо змінна x змінюється від –10 до 5 з кроком 0.5.

Розв'язання. Умовою задачі передбачається виконання обчислюваних процесів двох типів: циклічного та розгалуженого.

На першому кроці необхідно визначити: яка змінна обраховується першою – C чи D.

Далі необхідно розуміти, що значення змінних D та C залежать від змінної x, яка набуває значення в діапазоні від –10 до 5 з кроком 0,5, тому при складанні алгоритму потрібно організувати цикл. Лічильником циклу є змінна x, початкове значення лічильника дорівнює –10, кінцеве дорівнює 5, крок зміни дорівнює 0,5, умовою виходу з циклу є перевищення лічильником свого кінцевого значення.

Усередині циклу по змінній x використовується блок-схема розгалуженого процесу для визначення формули, за якою потрібно розраховувати значення змінної D.

На наступному кроці в циклі залежно від отриманого значення D обчислюється значення C.

Блок-схема алгоритму для наведеної задачі зображено на рисунку 4.8.

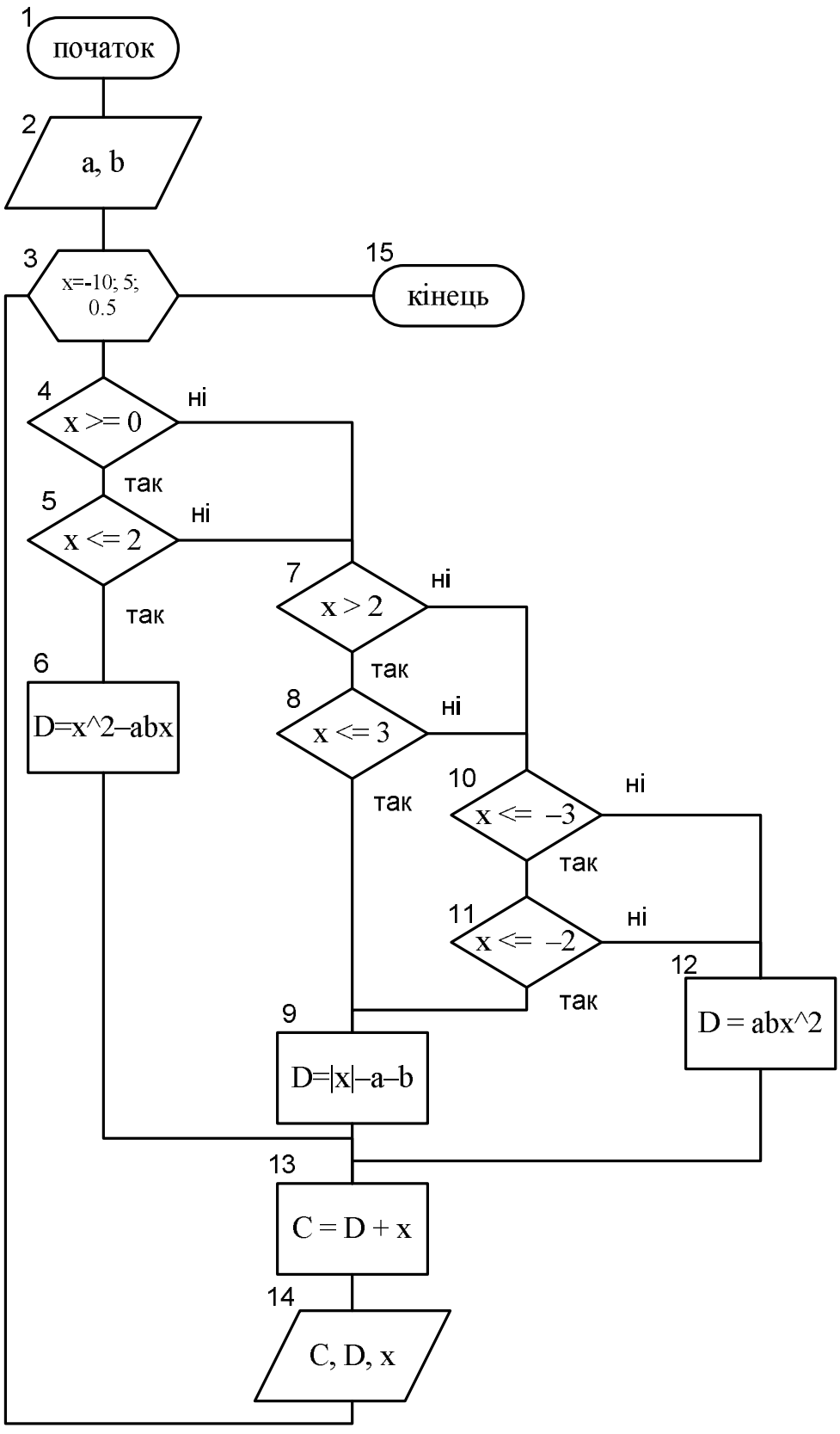


Рисунок 4.8 – Блок-схема алгоритму циклічного та розгалуженого обчислюваного процесу

В таблиці 4.1 наведені завдання до практичних занять та самостійного вивчення теми «Побудова циклічних алгоритмів».

Таблиця 4.1 – Варіанти індивідуальних завдань для побудови алгоритму циклічного обчислювального процесу

Номер варіанта	Теоретичне питання	Практичне завдання: скласти схему алгоритму
1	2	3
1	Який алгоритм називають циклічним?	$R = \frac{k!+1}{t+2k} + \prod_{i=1}^{5} i^3 + \frac{t^k}{2\sum_{i=1}^{3}(i+a)^2}$ Вивести на друк значення змінної R
2	Дати визначення поняттю «вкладені цикли»	$Z = \frac{2m!+3k!+\sqrt{m!+k!}}{\prod_{i=1}^{m}(i+m)^2 + 5\sum_{i=3}^{7}(m+k+i)^3}$ Вивести на друк значення змінної Z
3	Охарактеризувати властивості алгоритмів	$K = \begin{cases} \ln x_1, & \text{якщо } 0 < x_1 \le 3, \\ \cos x_1 + a, & \text{якщо } -3 < x_1 \le 0 \text{ або } x_1 = -3.5, \\ tg\, x_1 + b, & \text{якщо } x_1 = 4 \text{ або } x_1 = 5. \end{cases}$ $Q = Kx_1$, якщо $x_1 \in -10.1 \ldots 10.1$ із кроком 0.05. Вивести на друк значення змінних K, Q
4	Що собою являє ітераційний цикл?	$L = \frac{1}{2x} + \frac{2}{m!}\frac{(m+2)!}{2+a+x}$, $S = x\prod_{i=10}^{20}(i-m)^2 + 3{,}45L$, $x \in [0.1; 10]$, $\Delta x = 0.4$. Вивести на друк значення змінних L, S, x
5	Охарактеризувати типи алгоритмів	$Q = \frac{a+x^5}{(m+2)!} + \sqrt{\frac{m!}{x+3}} + \ln(m!)$, $x \in [1; 5]$, $\Delta x = 0.2$. Вивести на друк значення змінних Q, x

Продовження таблиці 4.1

1	2	3
6	Що таке параметр циклу?	$R = \frac{x^2 + \lg(m!+3k!)}{\sqrt[3]{x + m!+k!+a^x}},$ $x \in [3.2;\ 6.1],\ \ \Delta x = 0.1.$ Вивести на друк значення змінних *R, x*
7	Охарактеризувати циклічний алгоритм	$S = \frac{2.31y\left(\prod_{i=1}^{11}(i+в)\right)^2 + 3x}{\sum_{j=1}^{3}(j+mc) + 2.5x^2},$ $x \in [3;\ 12],\ \Delta x = 0.5.$ Вивести на друк значення змінних *x, S*
8	Який цикл називається арифметичним?	$Z = \frac{(m!+2k!) - x\sqrt{2k+1}}{\prod_{i=1}^{3}(i+2)^2 + \ln(m!+4) + e^x},$ $x \in [m; k],\ \Delta x = 0.2m.$ Вивести на друк значення змінних *x, Z*
9	Охарактеризувати типи алгоритмів	$Q = \frac{mb^2 + 2x}{\prod_{i=1}^{7}(i+2)} - \frac{\sqrt{k! + m!}}{m + 5x},$ $Z = \frac{k + m + 2q}{m + c + k} + 2,$ $x \in [1; 5],\ \Delta x = 0.2.$ Вивести на друк значення змінних *Q, Z, x*
10	Охарактеризувати розгалужений алгоритм	$F = \frac{\sqrt[3]{(m+a)!} + 2.2x^2 + \sin x}{\lg 2x + \lg 3x},$ $Z = \sqrt{f} + \sum_{i=3}^{9}(i + a^2) + \sqrt{x},$ $R = a + z - 13.5x$ $x \in [2; 9],\ \ \Delta x = 0.1.$ Вивести на друк значення змінних *F, R, Z, x*

Продовження таблиці 4.1

1	2	3
11	Які типи алгоритмів Вам відомі?	$y = \dfrac{\sqrt{m!+2k} + \prod_{i=2}^{5}(i+2k)^2}{3\sum_{i-1}^{7}(a^i + 2a + k!) + 2x}$ $x \in [3;\ k],\quad \Delta x = 0.5k.$ Вивести на друк значення змінних *x, y*
12	Пояснити призначення вкладених циклів	$Q = \dfrac{mb^2 + 2x}{\prod_{i=1}^{7}(i+2)} - \dfrac{\sqrt{k!+m!}}{m+5},$ $Z = \dfrac{k+m+2q}{m+c+k!} + 2^x,$ $x \in [a; c],\ \Delta x = 0.15a.$ Вивести на друк значення змінних *Q, Z, x*
13	Навести типи структур алгоритму	$R = \dfrac{k!+1}{t+2k} + \prod_{t_1=1}^{5} 3t_1 + \dfrac{t^k}{2\sum_{i=1}^{3}(i+a)^2},$ $t \in [12;15],\ \Delta t = 0.5.$ Вивести на друк значення змінних *R, t*
14	Що таке лічильник циклу?	$Q = \dfrac{x\sqrt{m+k!} + a\ln(m!+k)}{2x + \sin x^2 + \cos m},$ $x \in [7;17],\quad \Delta x = 0.7.$ Вивести на друк значення змінних *x, Q*
15	Пояснити на прикладі змінення лічильника циклу в заданих межах	$R = \sqrt{\prod_{i=2}^{9} \dfrac{i+a}{2} + 3f^3},$ $Z = \ln(a+R) + \log\left(\sum_{i=3}^{5} i^3\right),$ $Q = R + fZ - 5a$ $f \in [1;10],\quad \Delta f = 0.2a.$ Вивести на друк значення змінних *R, Z, Q, f*

Продовження таблиці 4.1

1	2	3
16	Охарактеризувати типи структур алгоритмів	$y = \dfrac{l! + n!}{\prod\limits_{i=1}^{3} i^4 + \sum\limits_{i=1}^{n} \dfrac{i^2 + a}{i + 10}}$ Вивести на друк значення змінної y
17	Охарактеризувати лінійний алгоритм	$A = \dfrac{k! + \sum\limits_{i=1}^{10} (i + a)}{k + 3} + \dfrac{\prod\limits_{j=1}^{3} j^3}{m!} + \dfrac{k}{2\sum\limits_{i=1}^{4} (i + 4)^2}$ Вивести на друк значення змінної A
18	Охарактеризувати типи циклів	$y = \dfrac{k^{x+1}}{\prod\limits_{i=1}^{3} i^2 + x} + \dfrac{m! + x^5 + (m+2)!}{k! + 2x + 3\sum\limits_{i=2}^{7} i^5},$ $x \in [4; 7], \quad \Delta x = 0.25.$ Вивести на друк значення змінних x, y
19	Які типи алгоритмів Вам відомі?	$P = \dfrac{(t+1)!}{\sum\limits_{i=1}^{k} i^2} + \prod\limits_{i=1}^{m} i^3 + \dfrac{k!}{m!}$ Вивести на друк значення змінної P
20	Охарактеризувати розгалужений алгоритм	$D = \dfrac{\sum\limits_{k=1}^{5} (k^2 + a)}{t + k} + \sum\limits_{i=1}^{n} \dfrac{i^2}{n} + k! + k!$ Вивести на друк значення змінної D
21	Дати визначення циклічного процесу	$S = \dfrac{m!}{\sum\limits_{i=1}^{m} \dfrac{i}{a + i}} + \dfrac{\prod\limits_{j=1}^{k!} j^2}{k!} + d!$ Вивести на друк значення змінної S
22	Дати визначення розгалуженого процесу	$D = \dfrac{\sum\limits_{i=15}^{25} (i^2 + a)}{\prod\limits_{j=1}^{30} (j^3 + b)},$ Вивести на друк значення змінної D

Продовження таблиці 4.1

1	2	3
23	Дати визначення лінійного процесу	$P = \frac{1}{k!} + \frac{3}{x} + \frac{(m+k)!}{x+a},$ $S = x\sum_{i=1}^{13}(i-m)^2 + 17.5P,$ $x \in [1;11], \quad \Delta x = 0.7.$ Вивести на друк значення змінних *P, S, x*
24	Надати визначення поняття «лічильник циклу»	$D = \frac{(a+b)!}{P_{10}+a} + \frac{\sum_{i=1}^{7} i^4}{\prod_{i=1}^{5}(i+a)^2}$ Вивести на друк значення змінної *D*
25	Надати визначення циклічного алгоритму	$D = \frac{b!+2}{2b+2} + \sum_{i=1}^{5}(i+a) + \frac{\prod_{i=1}^{6} j^4}{m!}$ Вивести на друк значення змінної *D*
26	Навести приклад вкладеного циклу	$A = \frac{x \cdot m!}{\sum_{i=5}^{15}(i+m)} + \frac{x^2}{k!},$ $x \in [0.1;\ 3.1] \quad \Delta x = 0.1.$ Вивести на друк значення змінних *A, x*
27	Надати визначення розгалуженого алгоритму	$P = \frac{m!}{k!} + \frac{(a+b)!}{\sum_{i=1}^{5}(i^2+a+b)} + \frac{\sum_{i=1}^{k!}(i+1)}{k!}.$ Вивести на друк значення змінної *P*
28	Охарактеризувати властивості алгоритму «результативність»	$Q = \frac{t!}{k!} + \frac{k!}{\sum_{i=1}^{n}(i^2+k^2)} + \frac{\prod_{j=1}^{n}(j+t)}{t!},$ Вивести на друк значення змінної *Q*
29	Надати визначення циклічного алгорит-му	$C = \frac{(a+b)!+\sum_{i=1}^{a!} i^2}{\prod_{i=1}^{k} j^3 + k!},$ Вивести на друк значення змінної *C*

Закінчення таблиці 4.1

1	2	3
30	Охарактеризувати властивості алгоритму «масовість»	$M = \begin{cases} a + bx, & якщо \quad 0 < x \le 5.5, \\ \log x, & якщо \quad 6 < x \le 7, \\ a - bx, & якщо \quad x < 0 \quad або \quad x = 10, \\ abx, & в \quad інших \quad випадках. \end{cases}$ $P = mx,$ $x \in [-0.8; 10], \quad \Delta x = 0.25$ Вивести на друк значення змінних *M, P, x*

Контрольні питання

1. Який обчислюваний процес називається циклічним?
2. Дати визначення циклу.
3. Що таке лічильник циклу?
4. Що таке параметр циклу?
5. Що потрібно визначити для організації циклу?
6. Що може бути умовою виходу з циклу?
7. Який цикл називається ітераційним?
8. Який цикл називається арифметичним?
9. Що таке вкладений цикл?
10. Який цикл називають внутрішнім циклом?
11. Який цикл називають зовнішнім циклом?
12. Навести блок-схему для підрахунку суми.
13. Навести блок-схему для підрахунку факторіалу.
14. Навести блок-схему для підрахунку добутку.
15. Скільки вхідних та вихідних потоків має блок циклу? Назвати їх.
16. Яка інформація записується в блоці циклу?
17. Які блоки замінює блок циклу?
18. Навести приклад розгалуженого процесу.
19. Навести приклад лінійного процесу.
20. Навести приклад циклічного процесу.
21. У яких випадках доцільно використовувати вкладені цикли і чому?

5 АЛГОРИТМІЗАЦІЯ У ОДНОВИМІРНИХ МАСИВАХ

Масив – це набір однотипних компонентів (елементів), що розташовані в пам'яті один за одним, доступ до яких відбувається за номером (індексом).

Розмірність масиву визначається кількістю індексів, якої достатньо для однозначного доступу до елемента масиву. Так, в одномірному масиві достатньо одного індексу для однозначної ідентифікації певного елемента масиву.

Масиви бувають динамічними та фіксованими.

Динамічним називається масив, кількість елементів якого може змінюватися під час виконання програми, в іншому випадку масив називається фіксованим.

Процес уведення елементів одномірного масиву складається з операцій зчитування та запису в пам'ять кожного з елементів один за одним. Такий процес повторення однакових операцій над різними даними зручно організовувати у вигляді циклу.

Якщо кількість елементів масиву є заздалегідь відомою, тобто масив є фіксованим, то лічильником циклу є індекс елемента масиву, початковим значенням лічильника циклу є значення 1 або 0, кінцевим значенням – кількість елементів масиву, кроком зміни лічильника є значення 1, умовою виходу з циклу – перевищення лічильником свого кінцевого значення. У цьому випадку тілом циклу є операція введення елемента, але для виведення масиву та проведення розрахунків у ньому організується такий самий цикл, але з іншими операціями в тілі циклу [6].

Приклади виконання задач.

Задача 1. Побудувати блок-схему алгоритму для підрахунку добутку ненулевих елементів одномірного масиву.

При роботі з одновимірним масивом необхідно розуміти, що вводяться однотипні елементи, які об'єднані одним ім'ям (в цьому випадку – це ім'я *A*), і які розрізняються індексами. При обробці елементів одновимірного масиву необхідно ввести в циклі всі елементи масиву, а потім вирішувати запропоновану задачу.

Розв'язання. Блок-схему алгоритму для підрахунку добутку ненулевих елементів одномірного масиву наведено на рисунку 5.1.

При розробці алгоритму необхідно:

– ввести у другому блоці кількість елементів масиву *N*;

– задати у третьому блоці змінній, яка міститиме результат добутку, початкове значення, яке не змінить кінцевого результату (для операції добутку це значення 1);

– організувати у наступному, четвертому, блоці цикл відносно змінної *i*, яка позначатиме номер елемента масиву;

– скласти тіло масиву з таких блоків: блок введення елементів масиву (5), блок перевірки елемента на рівність його нулю (6), блок зміни значення добутку

(7) – у випадку, якщо умова виконується, в інакшому випадку в циклі обробляється наступний елемент масиву з перевіркою на значення нулю.

Після завершення циклу остаточне значення добутку виводиться в блоці (8).

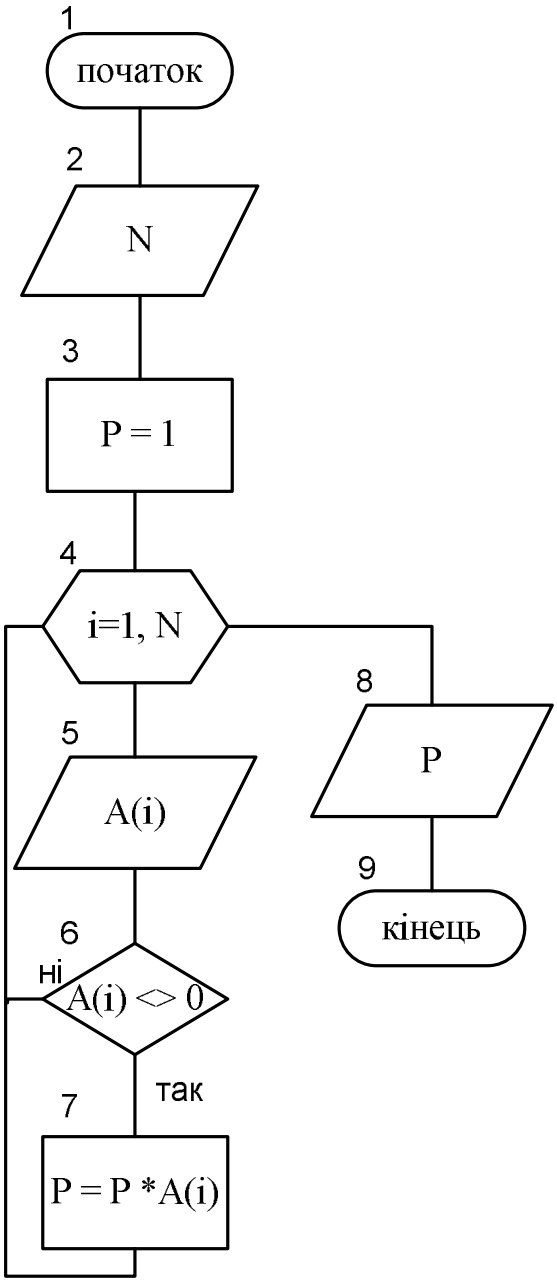


Рисунок 5.1 – Блок-схема алгоритму для підрахунку добутку ненулевих елементів одномірного масиву

Задача 2. Побудувати блок-схему алгоритму відшукання максимального елемента одномірного масиву.

Розв'язання. Блок-схему алгоритму пошуку максимального елемента одномірного масиву наведено на рисунку 5.2.

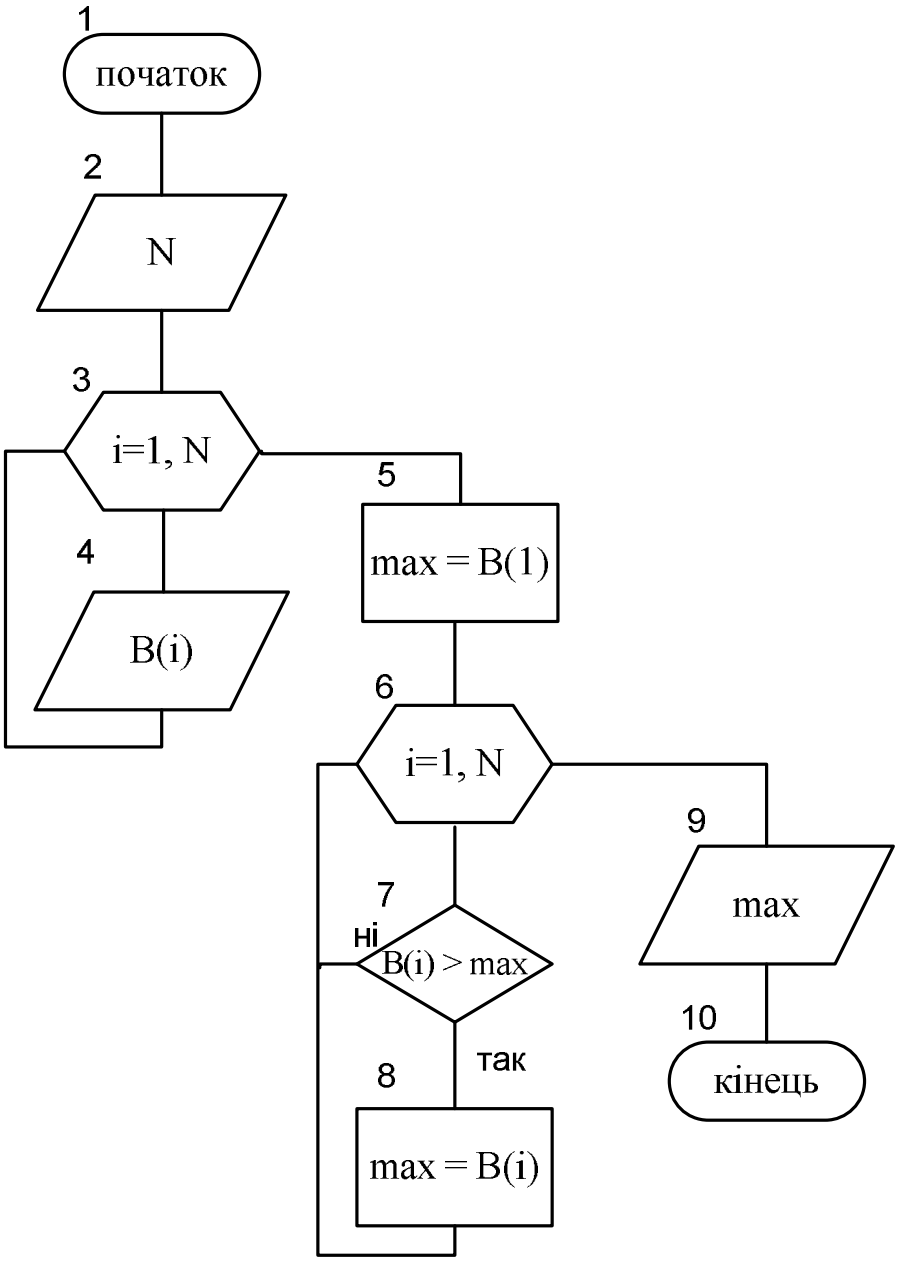


Рисунок 5.2 – Блок-схема алгоритму відшукання максимального елемента одномірного масиву

Для цього випадку організовано два окремих цикли: перший – для введення елементів масиву, другий – для перебирання всіх елементів та обрання елемента з найбільшим значенням.

Кінцевий результат, тобто максимальний елемент масиву буде міститися у змінній *max*. Перш ніж організовувати цикл для його пошуку, потрібно надати цій змінній певне значення, з яким надалі буде порівнюватися кожен наступний елемент масиву. Логічно привласнити значення першого елемента масиву (блок 5).

Далі організуємо цикл, тіло якого становлять: умовний блок, який перевіряє, чи є поточний елемент більшим за попередній елемент з найбільшим

значенням (блок 7), та блок привласнення змінній *max* нового найбільшого значення у разі виконання умови блоку 7 (блок 8).

Уже поза циклом у блоці 9 виводимо кінцеве значення змінної *max*, тобто максимальне значення елементів одномірного масиву.

Задача 3. Побудувати блок-схему алгоритму для підрахунку суми від'ємних елементів одномірного масиву.

Розв'язання. Блок-схему алгоритму для підрахунку суми від'ємних елементів одномірного масиву зображено на рисунку 5.3.

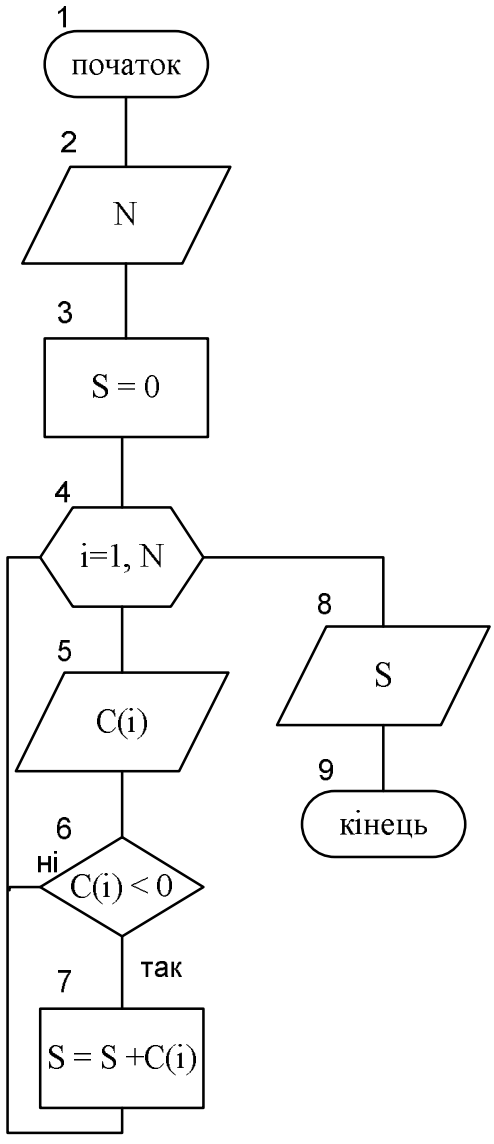


Рисунок 5.3 – Блок-схема алгоритму для підрахунку суми від'ємних елементів одномірного масиву

Блок-схема, зображена на рисунку 5.3 є аналогічною до блок-схеми, зображеної на рисунку 5.1. Змінилося значення, що присвоюється змінній результату в третьому блоці, оскільки результат додавання 0 не змінює значення

додатку; блок перевірки елемента, який є меншими за нулю; а також формула блоку 7 у тілі циклу.

Задача 4. Побудувати блок-схему алгоритму для підрахунку кількості нульових елементів одномірного масиву.

Розв'язання. Блок-схему алгоритму для підрахунку кількості нульових елементів одновимірного масиву наведено на рисунку 5.4.

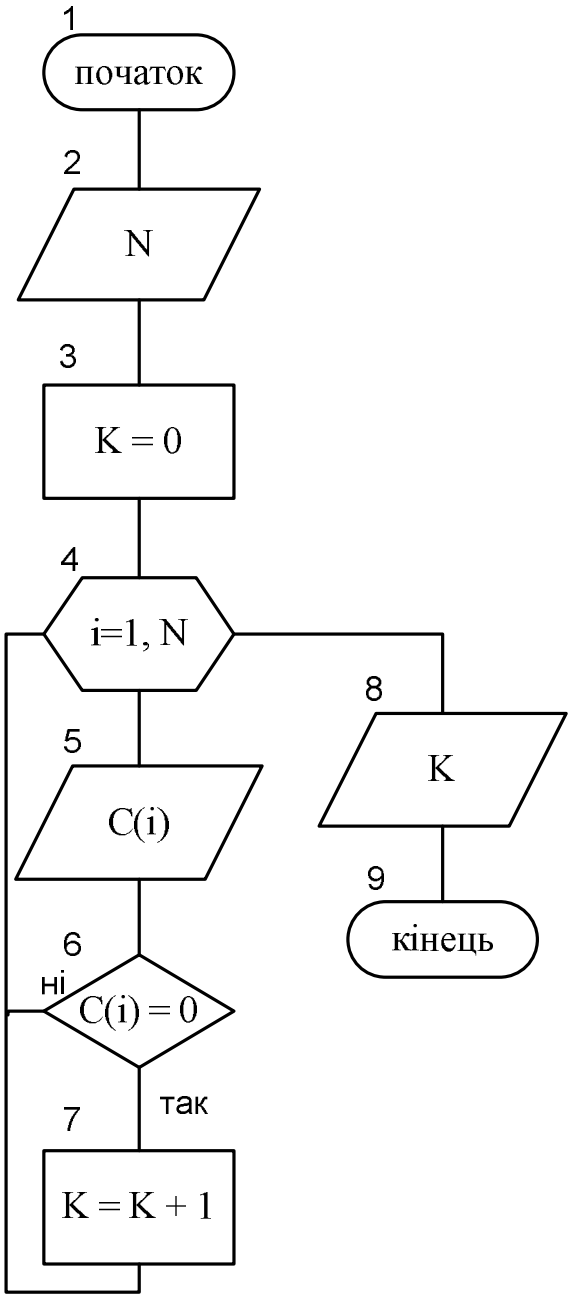


Рисунок 5.4 – Блок-схема алгоритму для підрахунку кількості нульових елементів одновимірного масиву

В таблиці 5.1 наведені завдання до практичних занять та самостійного вивчення теми «Алгоритмізація в одновимірних масивах».

Таблиця 5.1 – Варіанти індивідуальних завдань для обробки одномірного масиву

Номер варіанта	Зміст завдання
1	2
1	Визначити кількість елементів, не більших за задане число, яке дорівнює A = 12. Масив B(15)
2	Визначити кількість елементів, не менших за число A=7, що задане. Масив C(17)
3	Знайти номер елемента з максимальним значенням. Масив A(12)
4	Знайти номер елемента з мінімальним значенням. Масив D(14)
5	Знайти мінімальне число. Масив K(15)
6	Знайти максимальне число. Масив K(16)
7	Знайти додаток усіх елементів. Масив P(10)
8	Знайти добуток усіх елементів. Масив Z(11)
9	Знайти добуток елементів більших за 0,3. Масив P(10)
10	Знайти додаток елементів, які є меншими за 4. Масив Z(11)
11	Знайти добуток елементів, які є більшими за 3 та меншими за 3. Масив M(18)
12	Знайти додаток елементів, які є більшими за 25 або меншими за 10. Масив T(17).
13	Підрахувати кількість додатних елементів. Масив X(16)
14	Підрахувати кількість від'ємних елементів. Масив X(16)
15	Підрахувати кількість елементів, що дорівнюють нулю. Масив Y(17)
16	Знайти додаток додатних елементів. Масив Q(14)
17	Знайти додаток від'ємних елементів. Масив Q(14)
18	Знайти добуток від'ємних елементів. Масив I(19)
19	Знайти добуток додатних елементів. Масив I(19)
20	Визначити різницю між мінімальним та максимальним елементом. Масив A(20)
21	Визначити додаток елементів, розташованих на парних місцях. Масив A(20)
22	Визначити добуток елементів, розташованих на непарних місцях. Масив B(18)
23	Визначити різницю між добутками від'ємних та додатних елементів. Масив C(16)
24	Визначити кількість елементів, які є більшими за середнє арифметичне. Масив K(14)
25	Визначити кількість елементів, які є меншими за середнє арифметичне. Масив K(14)

Продовження таблиці 5.1

1	2
26	Визначити добуток елементів, які є більшими за 13 та меншими за 3. Масив D(16)
27	Визначити середнє арифметичне значення елементів масиву. Масив D(16)
28	Визначити добуток елементів, які є меншими за середнє арифметичне. Масив N(17)
29	Визначити додаток елементів, які є більшими за середнє арифметичне. Масив N(17)
30	Визначити середнє арифметичне значення додатних елементів. Масив Z(18)

Контрольні питання

1. Дати визначення масиву.
2. У чому полягає різниця між розмірністю та розміром масиву.
3. Охарактеризувати поняття «одновимірний масив». Навести приклад одновимірного масиву та провести паралель із математикою. Як у математиці називається одновимірний масив?
4. З яких операцій складається процес уведення елементів одномірного масиві?
5. У якому блоці доцільно виконувати введення числових значень елементів масиву?
6. Яким чином позначається блок циклу.
7. Описати поняття «лічильник циклу».
8. Якому значенню дорівнює кінцеве значення лічильнику циклу.
9. Як називається масив, у якому кількість елементів може змінюватися під час виконання програми?

6 СОРТУВАННЯ ВСТАВКАМИ І ВИБОРОМ. УДОСКОНАЛЕНЕ СОРТУВАННЯ ВСТАВКАМИ – СОРТУВАННЯ ШЕЛЛА

Алгоритм сортування – це алгоритм, що розв'язує задачу сортування, тобто здійснює впорядкування масиву елементів.

Термін сортування (англ. sorting) означає розділення елементів за певними ознаками (сортами) і не дуже точно описує поставлене завдання. Точнішою була б назва впорядкування (англ. ordering), але через перевантаженість слова «порядок» (англ. order) різними значеннями, при визначенні сортування цим терміном не скористалися.

Для алгоритму сортування (як і для будь-якого іншого сучасного алгоритму) основними характеристиками є такі:

1) час, який необхідний на впорядкування *n*-елементного масиву. Для значної кількості алгоритмів середній і найгірший час впорядкування *n*-елементного масиву є $O(n^2)$, і це пов'язано з тим, що в них передбачені перестановки елементів, що стоять поряд (різниця між індексами елементів не перевищує деякого заданого числа). Такі алгоритми зазвичай є стабільними, хоча й не ефективними для великих масивів. Інший клас алгоритмів здійснює впорядкування за час $O(nlogn)$. У цих алгоритмах використовується можливість обміну елементів, що розташовуються на будь-якій відстані один від одного;

2) необхідність додаткової пам'яті для сортування;

3) стабільність. Стабільне сортування не змінює взаємного розташування елементів з однаковими ключами.

Процес сортування перетворює масив, тобто замість первинного масиву отримуємо масив, елементи якого розташовано у порядку, що задовольняє певним умовам – умовам сортування.

Існує три простих методи сортування: сортування простими включеннями, сортування вибором та сортування обміном [3].

В алгоритмі *простими включеннями* (*сортування вставками*) використовується інкрементний підхід, за якого елементи умовно поділяються на готову послідовність: $a_1, a_2, \ldots a_{i-1}$, та вхідну послідовність: $a_i, \ldots a_n$.

На кожному кроці алгоритму обирається один з елементів вхідних даних і вставляється на потрібну позицію у вже відсортованому списку доти, доки набір вхідних даних не буде вичерпаний. Метод обрання чергового елемента із вхідного масиву є довільним; використовуватися може практично будь-який алгоритм обрання. Зазвичай (із метою отримання стійкого алгоритму сортування), елементи вставляються за порядком їхньої появи у вхідному масиві.

Метод ґрунтується на такій гіпотезі: вважається, що перед розглядом запису *R[j]* попередні записи: *R[1]*, *R[2]*,...,*R[j–1],* уже впорядковані, а *R[j]* вставляється у відповідне місце. Сортування масиву починається з другого запису, ключ якого порівнюється з ключем першого запису, і, якщо впорядкованість порушена, то записи *R[1]* і *R[2]* переставляються. Потім ключ запису *R[3]* порівнюється з ключами записів *R[2]* і *R[1]*. Як тільки алгоритм виявляє, що *(j+1)*-й елемент масиву є меншим за *j*-й елемент (за сортування в порядку зростання), він копіює значення цього елемента в буферну змінну й з

початку масиву до *j* аналізує доти, доки значення буферної змінної не буде меншим за будь-який елемент *x*. Потім частина масиву, починаючи з *x* і закінчуючи *j*, переміщується на одну позицію в бік зростання, і на місце, що утворилося, записується значення елемента, який переміщується. Далі продовжується переміщення основного масиву до елемента *n–1* (оскільки порівнюються *j*-й і *(j+1)*-й елементи).

У книзі Кормена Т. Х. «Алгоритми: побудова та аналіз» наведена така візуалізація роботи алгоритму вставками (рис. 6.1) [5].

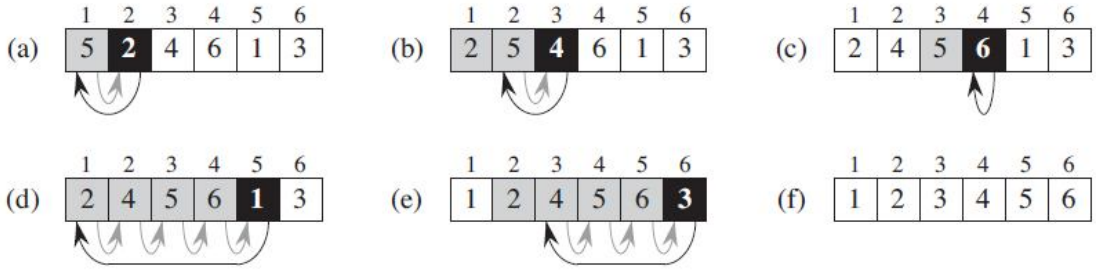


Рисунок 6.1 – Сортування *Insertion-Sort* для масиву з елементами 5, 2, 4, 6, 1, 3

Сортування вставками має такі переваги:

1) ефективне для невеликих наборів даних, для наборів даних до десятків елементів може виявитися ще ефективнішим;
2) ефективне для наборів даних, які вже частково відсортовані;
3) є стійким алгоритмом сортування (не змінює порядок елементів, які вже відсортовані);
4) може виконувати сортування списку в процесі його отримання;
5) може працювати значно швидше завдяки бінарному пошуку.

Недоліком сортування вставками є дуже висока обчислювальна складність алгоритму (в разі використання стандартного алгоритму).

Удосконаленим алгоритмом сортування простими включеннями є *сортування Шелла* [3]. Розглянемо масив із восьми елементів: 44, 55, 12, 42, 94, 18, 06, 67. Згідно з цим методом сортування на першому проході окремо групуються всі елементи, які стоять один від одного на чотири позиції. Цей процес називається 4-сортуванням (рис. 6.2)

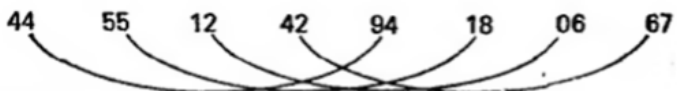


Рисунок 6.2 – Сортування Шелла – перший прохід

Після цього елементи знову об’єднуються в групи з елементами, які відстоять один від одного на дві позиції, і сортуються заново. Цей процес зветься 2-сортуванням (рис. 6.3).

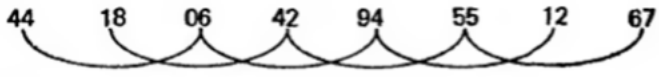


Рисунок 6.3 – Сортування Шелла – другий прохід

На третьому проході всі елементи сортуються звичайним сортуванням (рис. 6.4)

Рисунок 6.4 – Сортування Шелла – 1-сортування

У сортуванні Шелла на кожному кроці сортування беруть участь порівняно мало елементів, або вони вже достатньо добро впорядковані й потребують порівняно мало перестановок.

Метод *сортування простим вибором*, у певному сенсі є протилежним до методу сортування простими включеннями: під час сортування простими включеннями на кожному кроці розглядається тільки один черговий елемент вхідної послідовності та всі елементи готового масиву для знаходження місця включення. Під час сортування простим вибором розглядаються всі елементи вхідного масиву для знаходження елементів з найменшим ключем, і цей один наступний елемент відправляється до готової послідовності. Цей метод ґрунтується на такому правилі:

1) обирається елемент із найменшим ключем;

2) обраний елемент міняється місцем із першим елементом a_1.

Наведені операції повторюються з $n–1$ елементами, що залишилися, потім із $n–2$ елементами, поки не залишиться лише один елемент – найбільший.

Ідея методу полягає в створенні відсортованої послідовності шляхом приєднання до неї елементів одного за іншим в правильному порядку.

Наприклад, для побудови послідовності, починаючи з лівого кінця масиву будується алгоритм із n послідовних кроків, починаючи від нульового та закінчуючи $(n–1)$-им. На i-му кроці обирається найменший з елементів $a[i]$ ... $a[n]$. Обраний елемент міняється місцем з елементом $a[i]$ [3; 7; 8].

Найпростіший метод сортування полягає в поелементному переборі та почерговому порівнянні кожного елемента за певною ознакою (умовою сортування). Однак такий процес є не дуже ефективним із точки зору використання обчислювальних ресурсів, але прийнятним для сортування масивів з невеликою кількістю елементів.

Приклади розв’язання задач.

Задача 1. Побудувати блок-схему алгоритму для сортування вставками елементів одновимірного масиву [9] двома варіантами.

Розв’язання. Блок-схему алгоритму для сортування вставками елементів одновимірного масиву наведено на рисунку 6.5 (варіант 1).

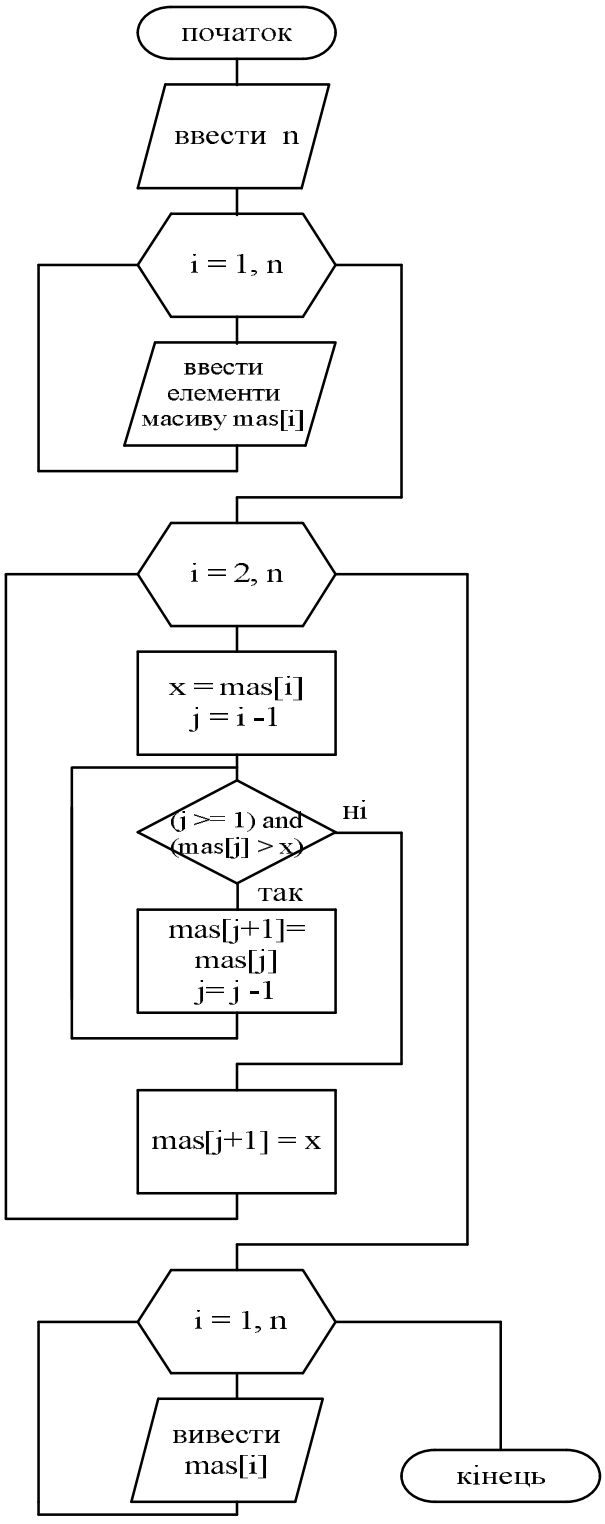


Рисунок 6.5 – Блок-схема алгоритму сортування вставками (варіант 1)

Розв'язання. Блок-схему алгоритму для сортування вставками елементів одновимірного масиву наведено на рисунку 6.6 (варіант 2).

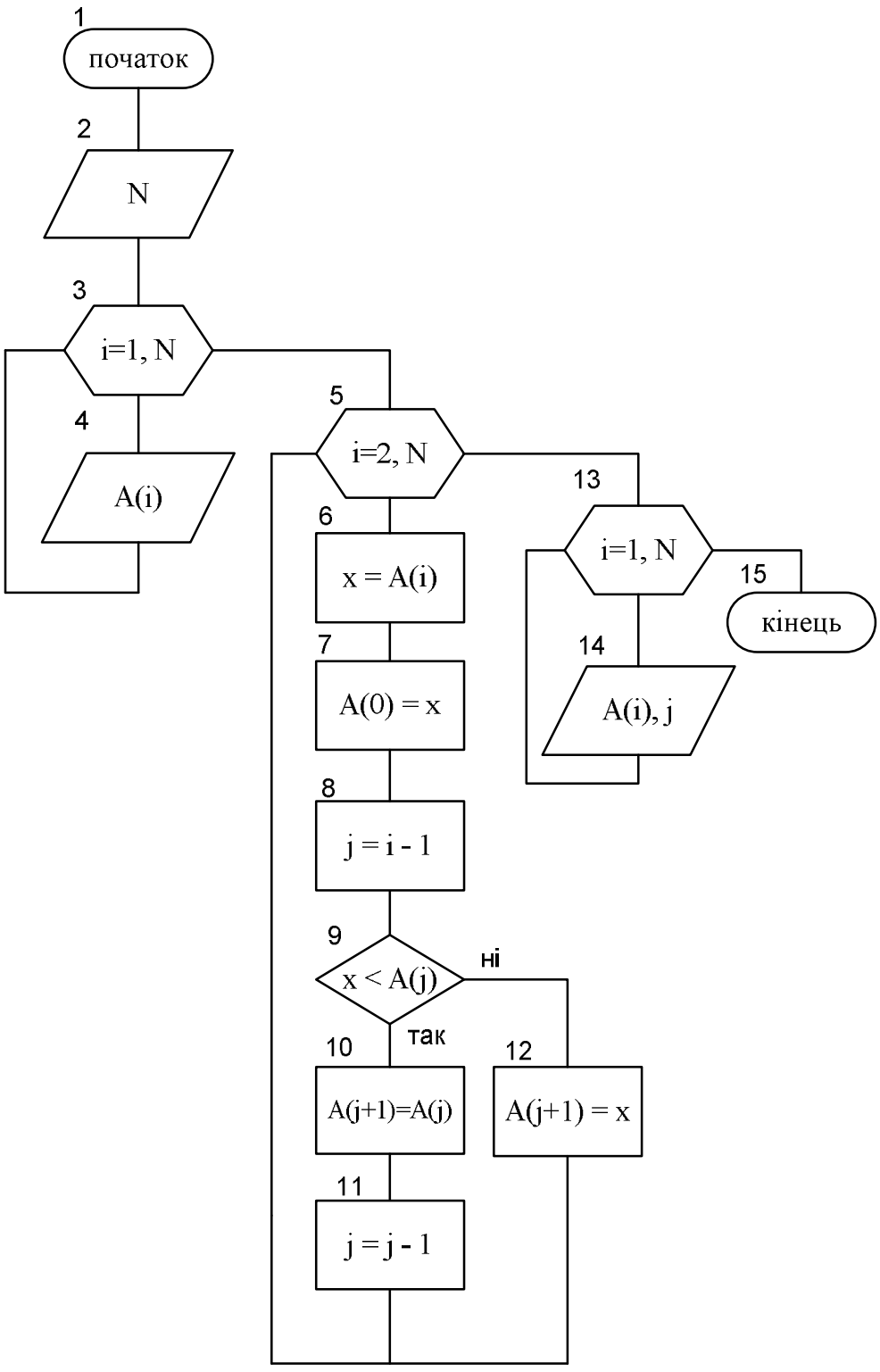


Рисунок 6.6 – Блок-схема алгоритму для сортування елементів одновимірного масиву вставками (варіант 2)

Задача 2. Побудувати блок-схему алгоритму для сортування елементів одновимірного масиву методом удосконаленого сортування вставками (простим включенням) – сортуванням Шелла (рис. 6.7).

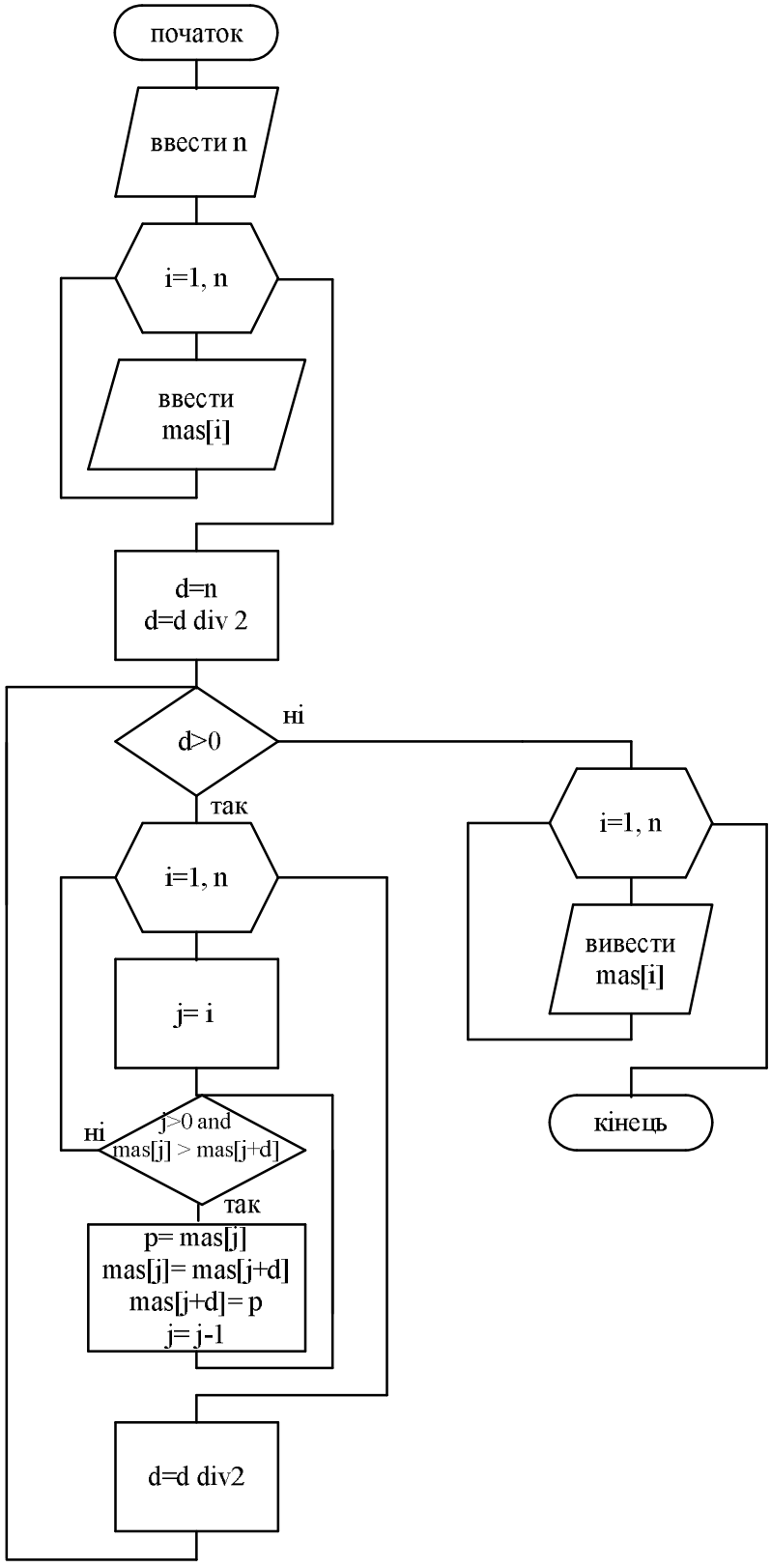


Рисунок 6.7 – Блок-схема алгоритму сортування Шелла елементів одновимірного масиву

Задача 3. Побудувати блок-схему алгоритму для сортування елементів одномірного масиву за убуванням методом вибору.

Розв'язання. Блок-схему алгоритму для сортування елементів одновимірного масиву за убуванням методом вибору наведено на рисунку 6.8.

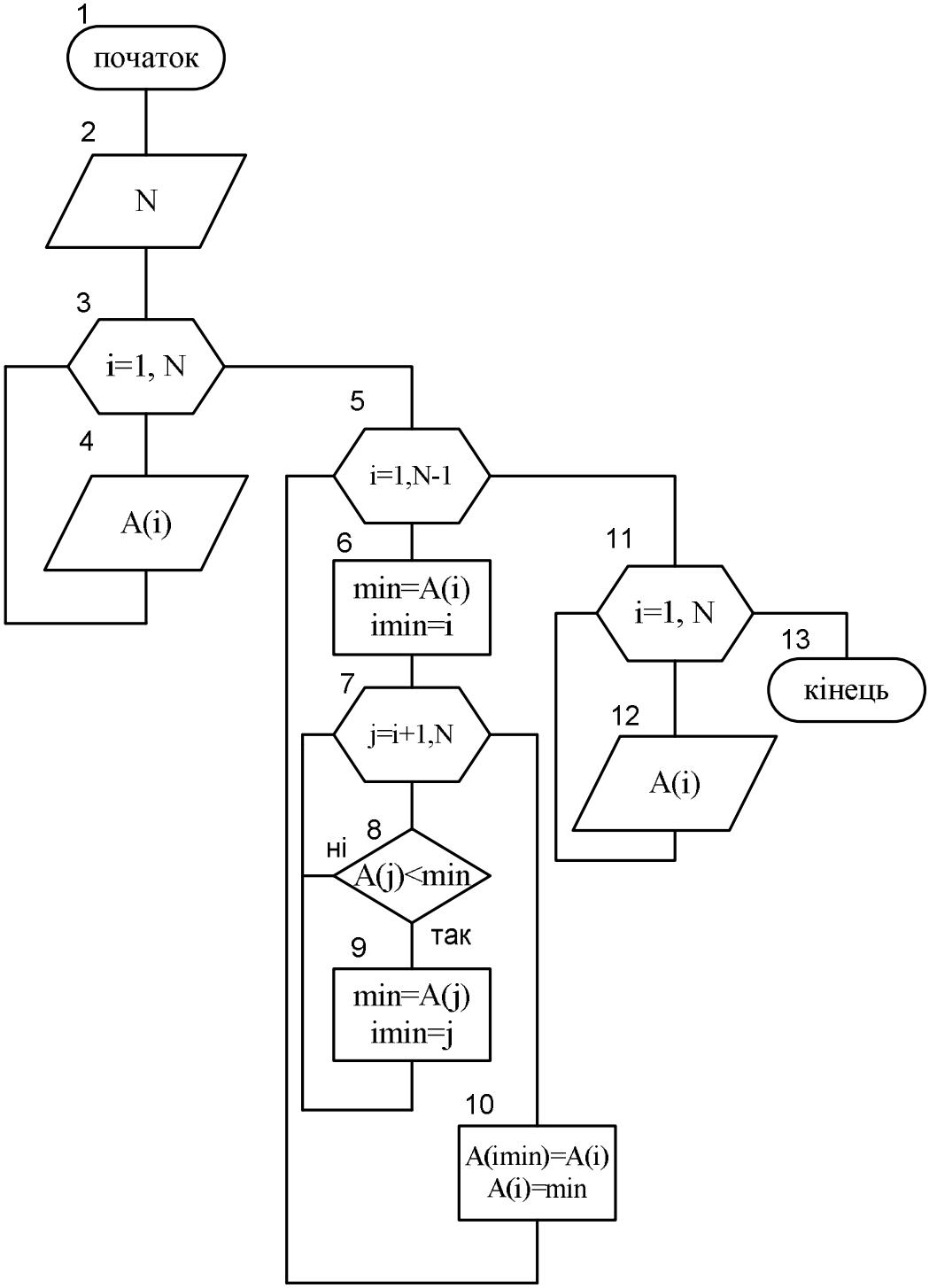


Рисунок 6.8 – Блок-схема алгоритму для сортування елементів одновимірного масиву за убуванням сортуванням вибором

В таблиці 6.1 наведені завдання до практичних занять та самостійного вивчення теми «Сортування вставками та вибором. Удосконалене сортування вставками – сортування Шелла».

Таблиця 6.1 – Варіанти індивідуальних завдань для сортування вставками та вибором

Номер варіанта	Завдання
1	Перетворити одномірний масив, що складається з n дійсних елементів, так, щоб спочатку були розташовані всі елементи, що відрізняються від максимального не більше ніж на 20 %, а потім – усі інші
2	Увести послідовність натуральних чисел $\{A_j\}$, $j = 1...n$ ($n <= 1000$). Упорядкувати послідовність у порядку спадання першої цифри числа
3	Виконати в масиві $A(n)$ (якщо $n <= 1000$) сортування у порядку зростання. Знайти серед елементів відсортованого масиву мінімальний елемент та його номер
4	Об'єднати в один масив два одновимірних масиви з різною кількістю елементів і натуральне число k. При цьому потрібно включити другий масив між k-им і $(k + 1)$-им елементами першого масиву, не використовуючи додатковий масив
5	Утворити нову послідовність чисел з двох послідовностей: $a_1 <= a_2 <= ... <= a_{n-1} <= a_n$ і $b_1 <= b_2 <= ... <= b_{n-1} <= b_n$, так, щоб вона була зростаючою. Додатковий масив використовувати не можна
6	Відсортувати в порядку спадання методом «бульбашок» одновимірний цілочисельний масив, заданий випадковими числами на проміжку (–100; 100). Вивести на екран початковий та відсортований масиви
7	Увести послідовність тризначних натуральних чисел $\{A_j\}$, $j = 1...n$ ($n <= 1000$). Упорядкувати послідовність у порядку спадання додатка двох перших цифр числа
8	Знайти медіану у масиві розміром $(2m + 1)$, який заповнено випадковим чином, де m – натуральне число. Медіаною називається елемент ряду, який поділяє його на дві рівні частини, в одній з яких розташовуються елементи, які є не меншими за медіану, а в іншій – не більшими за медіану
9	Перетворити одномірний масив, що складається з n дійсних елементів, так, щоб у першій його половині були розташовані елементи, які стоять на парних позиціях, а в другій – на непарних -

Продовження таблиці 6.1

1	2
10	Увести послідовність натуральних чисел $\{A_j\}$, $j = 1...n$ ($n <= 1000$). Упорядкувати послідовність у порядку спадання суми цифр числа.
11	Переставити елементи в послідовності чисел $a_1, a_2, \ldots a_n$ так, щоб вони були розташовані у порядку спадання. Для цього в масиві, починаючи з першого, обирається найбільший елемент і ставиться на перше місце, а перший – на місце найбільшого. Потім, починаючи з другого елемента, ця процедура повторюється. Скласти алгоритм сортування методом обрання
12	Відсортувати у порядку спаданням методом вибору одновимірний масив дійсних чисел, заданий випадковими числами на проміжку [0; 50]. Вивести на екран початковий та відсортований масиви
13	Знайти *моду* в масиві розміром m, який заповнено випадковим чином, де m – натуральне число. *Модою* називається елемент ряду, який зустрічається найчастіше
14	Перетворити одномірний масив, що складається з n дійсних елементів, таким чином, щоб спочатку були розташовані всі додатні елементи, а потім – усі інші (елементи, що дорівнюють 0, також потрібно вважати додатними)
15	Увести послідовність натуральних чисел $\{A_j\}$, $j = 1...n$ ($n <= 1000$). Упорядкувати послідовність у порядку спадання
16	Переставити числа в порядку зростання в послідовності чисел $a_1, a_2,...a_n$. Для цього порівнюються два сусідніх числа: a_n і a_{n+1}. Якщо $a_n > a_{n+1}$, то виконується перестановка. Так продовжується доти, доки всі елементи не виявляться розташованими в порядку зростання. Скласти алгоритм сортування методом обміну, підраховуючи при цьому кількість перестановок
17	Відсортувати у порядку зростання методом простого включення одновимірний цілочисельний масив, заданий з клавіатури різними числами. Вивести на екран початковий та відсортований масиви
18	Відсортувати у порядку зростання методом «бульбашок» одновимірний цілочисельний масив, заданий випадковими числами на проміжку [–90; 90]. Вивести на екран початковий та відсортований масиви
19	Перетворити одномірний масив, що складається з n дійсних елементів, так, щоб спочатку були розташовані елементи, що дорівнюють нулю, а потім – усі інші
20	Ввести послідовність тризначних натуральних чисел $\{A_j\}$, $j = 1...n$ ($n <= 1000$). Впорядкувати послідовність у порядку спадання

Закінчення таблиці 6.1

1	2
21	Переставити числа в послідовності чисел $a_1, a_2,...a_n$ в порядку зростання за допомогою сортування вставками Виконується сортування в такий спосіб. Нехай $a_1, a_2, ..., a_n$ – упорядкована послідовність, тобто $a_1 <= a_2 <= ... <= a_n$. Береться наступне число a_{n+1} і вставляється в послідовність так, щоб нова послідовність була також зростаючою. Процес виконується доти, доки всі елементи від $i + 1$ до n, не будуть перебрані
22	Відсортувати у порядку зростання методом простого включення одновимірний цілочисельний масив, заданий з клавіатури різними числами. Вивести на екран початковий та відсортований масиви
23	Відсортувати у порядку зростання методом вибору одновимірний масив дійсних чисел, заданий випадковими числами на проміжку [0; 50]. Вивести на екран початковий та відсортований масиви
24	Перетворити одномірний масив, що складається з n дійсних елементів, так, щоб у першій його половині були розташовані елементи, які стоять на непарних позиціях, а в другій – на парних
25	Увести послідовність тризначних натуральних чисел $\{A_j\}, j = 1...n$ ($n <= 1000$). Впорядкувати послідовність у порядку спадання третьої цифри числа
26	Утворити з двох заданих послідовностей $a_1 => a_2 => ... => a_n$ і $b_1 <= b_2 <= ... <= b_n$, нову послідовність чисел так, щоб вона теж була спадною
27	Відсортувати у порядку спадання методом простого включення одновимірний цілочисельний масив, заданий з клавіатури різними числами. Вивести на екран початковий та відсортований масиви
28	Перетворити одномірний масив, що складається з n дійсних елементів, в такий спосіб, щоб у першій його половині були розташовані елементи, які стоять на непарних позиціях, а в другій – на парних
29	Увести послідовність тризначних натуральних чисел $\{A_j\}, j = 1...n$ ($n <= 1000$). Упорядкувати послідовність за спаданням добутку цифр числа
30	Знайти в одномірному масиві дійсних чисел максимальний елемент. Відсортувати масив у такий спосіб, щоб перед цим елементом числа були розташовані у порядку зростання, а після нього – у порядку спадання

Контрольні питання

1. Сформулювати мету процесу сортування.
2. Описати алгоритм сортування методом простих вставок. Якими є переваги та недоліки даного методу сортування.
3. У чому полягає різниця між сортуванням методом простих вставок та сортуванням методом вибору?
4. На якому правилі заснований метод сортуванням вибором?
5. Описати ідею алгоритму сортування вибором.
6. У чому полягає сортування обміном?
7. Описати алгоритм сортування методом «бульбашок».
8. Порівняти кожен із методів сортування та акцентувати увагу на величині вибірки, перевагах та недоліках кожного з методів.
9. У зв’язку з чим існує велика кількість алгоритмів сортувань?
10. З якою метою використовуються прості методи сортування, якщо для них є характерною низька ефективність?
11. У чому полягає відмінність між принципами сортування у порядку спадання від сортування у порядку зростання?
12. Застосування яких наборів первинних даних виявляє ефективність алгоритмів методів простих сортувань, порівнюючи їх один з одним?

7 СОРТУВАННЯ БУЛЬБАШКАМИ. УДОСКОНАЛЕНЕ СОРТУВАННЯ БУЛЬБАШКАМИ – ШЕЙКЕР-СОРТУВАННЯ

Існують інші методи сортування з використанням обміну, одним з яких є метод «бульбашок» і його удосконалена версія – шейкер-сортування.

Для опису основної ідеї *бульбашкового сортування* (*Bubble Sort*) необхідно записати ключі в масиві, які сортуються, розташованими вертикально. Алгоритм, що реалізує метод «бульбашок» полягає у здійсненні повторюваних проходжень по масиву, що сортується. За кожен прохід елементи послідовно порівнюються попарно і, якщо порядок у парі є невірним, виконується обмін елементів місцями. Проходження по масиву повторюються *N–1* разів (*N* – кількість елементів масиву) або доти, доки під час чергового проходження не виявиться, що обміни більше не потрібні, що означає – масив відсортований. Під час кожного проходження алгоритму із внутрішнього циклу черговий найбільший елемент масиву ставиться на своє місце в кінці масиву поряд із попереднім «найбільшим елементом», а найменший елемент переміщується на одну позицію до початку масиву («спливає» до потрібної позиції, як бульбашка на воді, звідси й походить назва алгоритму) [3; 8].

Єдина перевага бульбашкового сортування – це його простота. Bubble Sort є популярним, але неефективним алгоритмом сортування. Як в алгоритмі сортування простими вставками, так і в бульбашковому сортуванні в кожен момент часу елементи переміщуються тільки на одну позицію. Такі алгоритми потребують порядку n^2 операцій. Отже, удосконаленням такого сортування є обмін ключів, які розташовані далеко друг від друга, або використання швидкого сортування (практичне заняття 7).

Шейкер-сортування є ефективнішою формою бульбашкового сортування, яке ще називають: сортування перемішуванням, пульсуюче сортування, двонаправлене сортування «бульбашкою».

В алгоритмі шейкер-сортування на відміну від методу бульбашок змінюються напрями наступних один за іншим проходів.

При роботі цього сортування межі тієї частини масиву, в якій є перестановки, звужуються, крім того, внутрішні цикли проходять по масиву то в один, то в інший бік, піднімаючи найлегший елемент угору й опускаючи найважчий елемент у самий низ за одну ітерацію зовнішнього циклу. Отже, за одну ітерацію зовнішнього циклу мінімальне та максимальне значення займають свої позиції в масиві, що істотно впливає на час роботи алгоритму.

Задача 1. Побудувати блок-схему алгоритму для сортування елементів одномірного масиву методом «бульбашок».

Розв'язання. Блок-схему алгоритму для сортування елементів масивів методом «бульбашок» наведено на рисунку 7.1 (варіант 1).

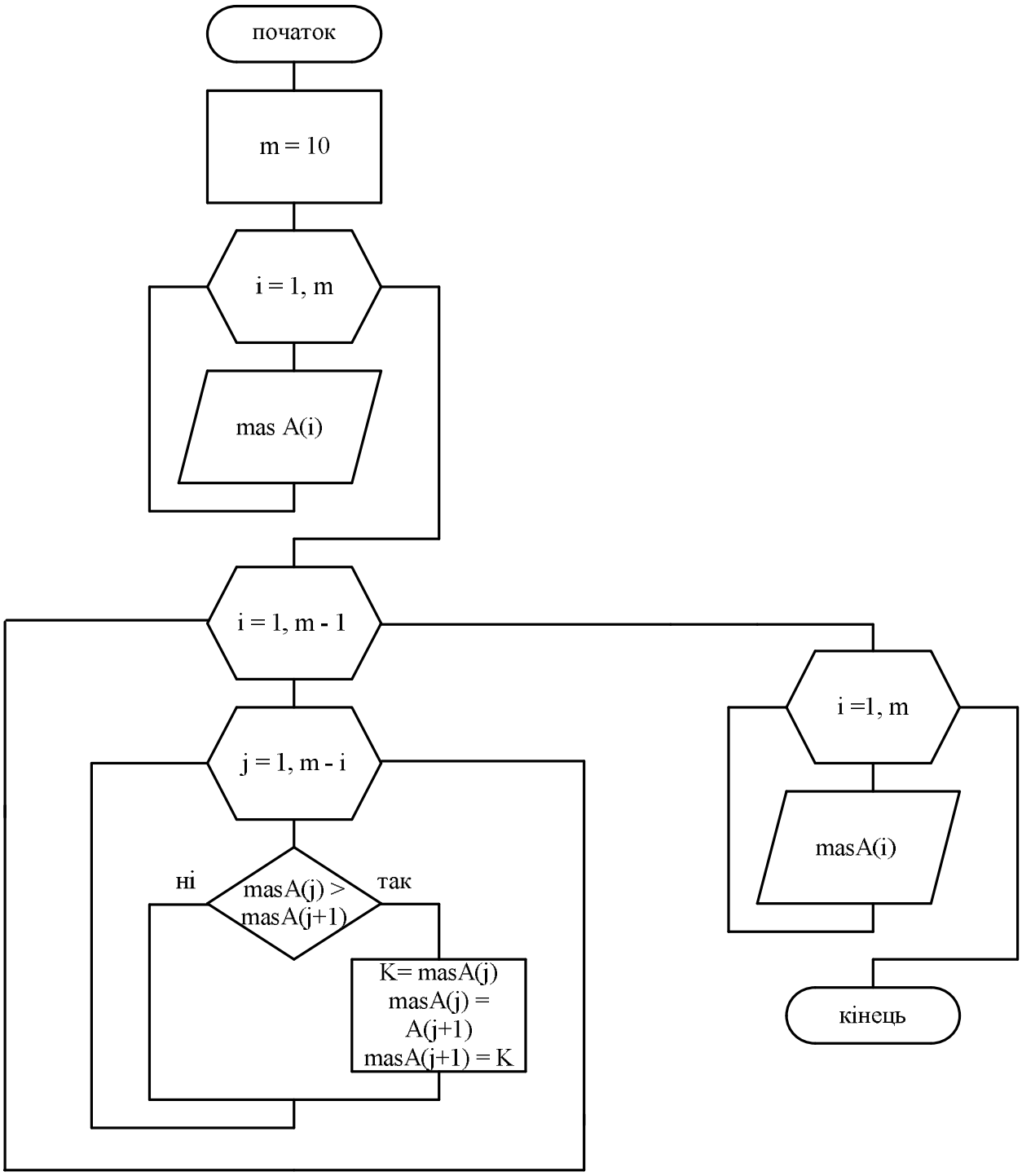


Рисунок 7.1 – Блок-схема алгоритму для сортування елементів масивів методом «бульбашок» (варіант 1)

Задача 2. Задано масив студентів F (до 30 елементів) та відповідний йому масив успішності кожного зі студентів (масив значень середнього бала кожного студента) з тією самою кількістю елементів.

Побудувати блок-схему алгоритму для сортування елементів масиву прізвищ студентів та масиву успішності студентів за убуванням методом «бульбашок».

Розв'язання. Блок-схему алгоритму для сортування елементів масивів методом «бульбашок» наведено на рисунку 7.2 (варіант 2).

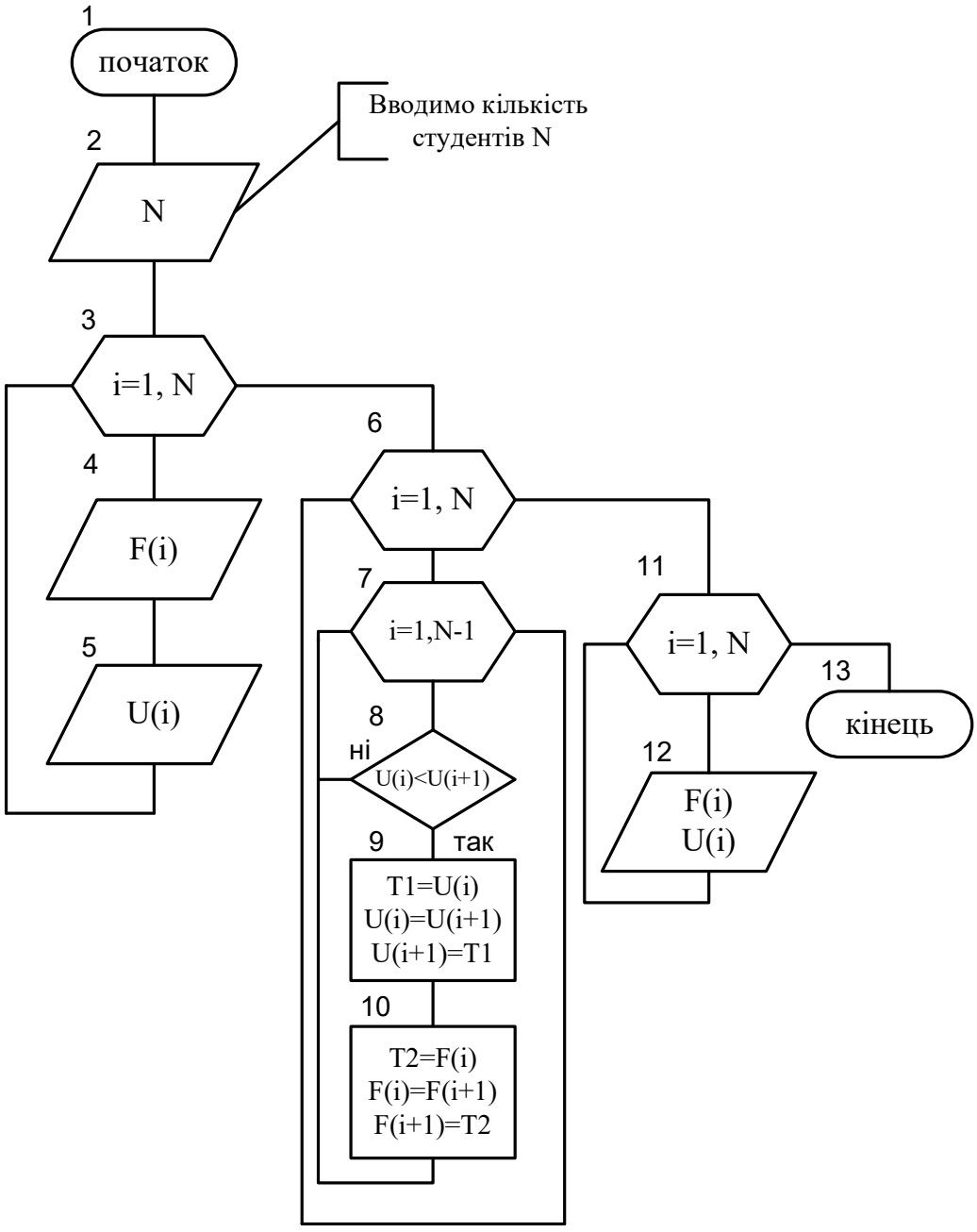


Рисунок 7.2 – Блок-схема алгоритму для сортування елементів масивів методом «бульбашок» (варіант 2)

Розв'язання. Блок-схему алгоритму для сортування елементів масивів методом «бульбашок» наведено на рисунку 7.3 (варіант 3).

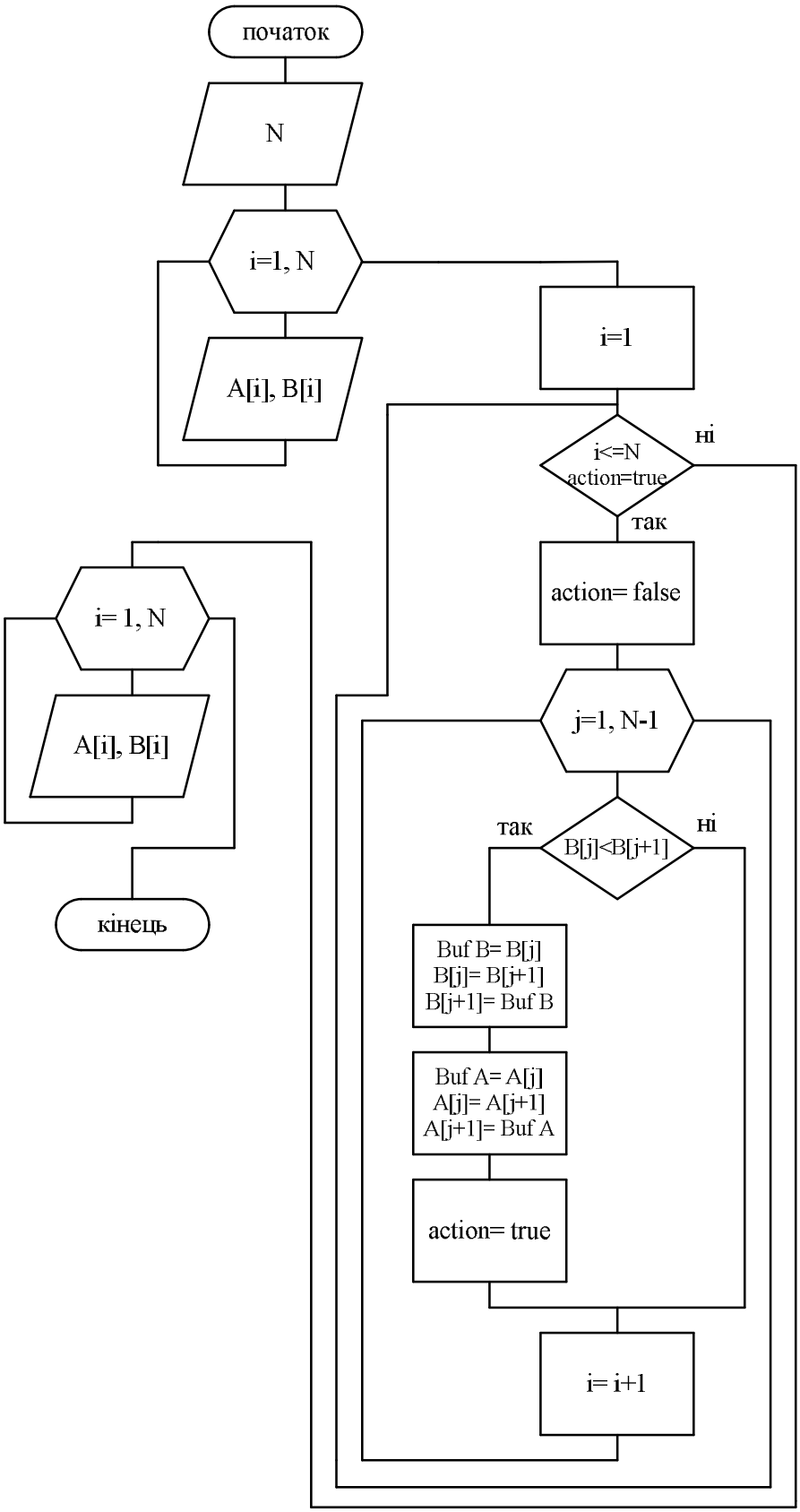


Рисунок 7.3 – Блок-схема алгоритму для сортування елементів масивів методом «бульбашок» (варіант 3)

Задача 3. Побудувати блок-схему алгоритму для сортування елементів одномірного масиву методом шейкер-сортування (рис. 7.4).

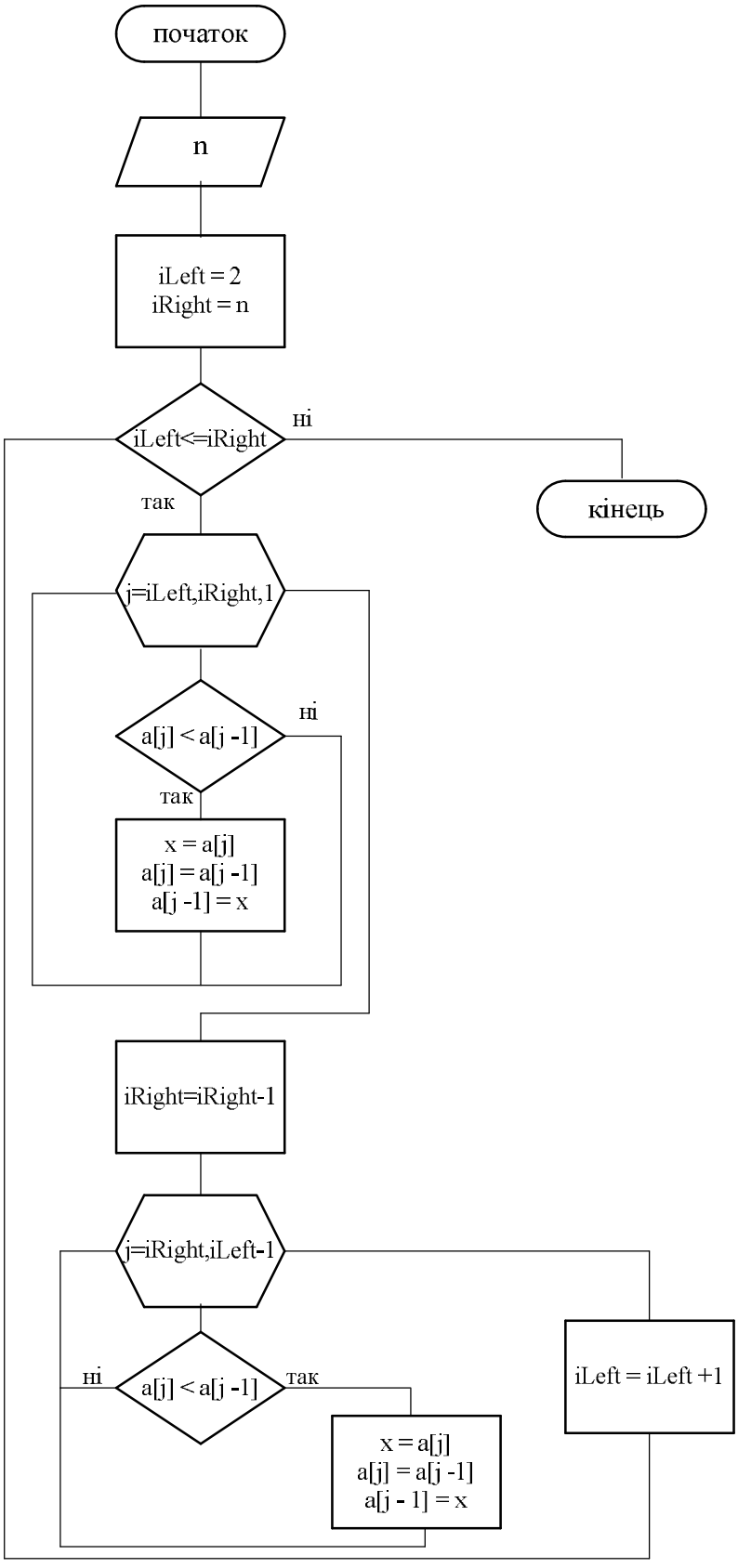


Рисунок 7.4 – Блок-схема алгоритму для сортування елементів масивів шейкер-сортуванням

Для відпрацювання сортування бульбашками та шейкер-сортування використати завдання, які наведено в таблиці 5.1.

Контрольні питання

1. Сформулювати мету процесу сортування.
2. У чому полягає сортування обміном?
3. Описати алгоритм сортування методом «бульбашок».
4. Порівняти кожен із методів сортування та акцентувати увагу на величині вибірки, перевагах та недоліках кожного з методів.
5. У зв’язку з чим існує велика кількість алгоритмів сортувань?
6. З якою метою використовуються прості методи сортування, якщо їм властива низька ефективність?
7. Охарактеризувати ідею бульбашкового сортування.
8. Охарактеризувати шейкер-сортування.
9. У чому полягає перевага методу шейкер-сортування порівняно з методом бульбашкового сортування?

8 ШВИДКЕ ТА ПІРАМІДАЛЬНЕ СОРТУВАННЯ

Швидке сортування (*Quick Sort*) – алгоритм сортування, добре відомий, як алгоритм розроблений в 1960 році Чарльзом Ентоні Річардом Хоаром (C. A. R. Hoare), який розробив цей метод стосовно до машинного перекладу, працюючи над розробкою російсько-англійського розмовника. Словник зберігався на магнітній стрічці, і сортування слів тексту, що оброблявся, дозволяло отримати їх переклади за один прогін стрічки, без перемотування її назад.

Quick Sort є істотно поліпшеним варіантом алгоритму сортування за допомогою прямого обміну, варіантами якого є бульбашкове сортування та шейкер-сортування. Відомо, що алгоритми обміну характеризуються низькою ефективністю. Принципова відмінність швидкого сортування полягає в тому, що найперше проводяться перестановки на найбільшій можливій відстані й після кожного проходу елементи діляться на дві незалежні групи. Цікаво, що поліпшення самого неефективного прямого методу сортування дало в результаті один із найефективніших поліпшених методів: сортування Хоара не потребує додаткової пам'яті й виконує у середньому $O(n\ log\ n)$ операцій, а у найгіршому випадку робить $O(n^2)$ порівнянь. Оскільки алгоритм використовує дуже прості цикли й операції, він працює швидше інших алгоритмів, що мають таку саму асимптотичну оцінку складності. Швидке сортування є алгоритмом на основі порівнянь, і не є стабільним.

При загальному розгляді алгоритму швидкого сортування, необхідно відзначити, що цей метод ґрунтується на послідовному поділі набору даних, який сортується, на блоки меншого розміру так, що між значеннями різних блоків забезпечується відношення впорядкованості (для будь-якої пари блоків усі значення одного з цих блоків не перевищують значень іншого блоку).

Опорним (*pivot*) елементом називається елемент масиву, який обирається в певний спосіб. З точки зору коректності алгоритму вибір опорного елемента байдужий. З точки зору підвищення ефективності алгоритму обиратися повинна медіана, але без додаткових відомостей про дані, які сортуються, її зазвичай неможливо отримати. Оскільки необхідно обирати постійно один і той самий елемент, то в алгоритмі Хоара обирають або середній, або останній по положенню елемент масиву [10–12].

Розглянемо візуалізацію роботи сортування Хоара на прикладі масиву елементів, наведених на рисунку 8.1.

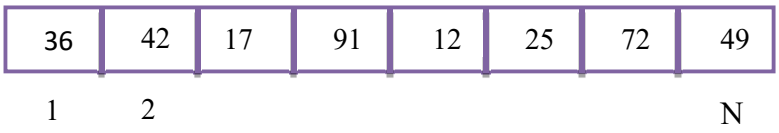


Рисунок 8.1 – Початковий масив

Візьмемо за опорний елемент перший елемент масиву *A[1]* і порівняємо його з останнім елементом *A[N]*. Якщо ці елементи впорядковано, то *A[1]*

порівнюємо з *A[N–1], A[N–2],* і так буде виконуватися, поки не зустрінеться елемент, менший від опорного елементу. У такому випадку ці елементи потрібно поміняти місцями (рис. 8.2).

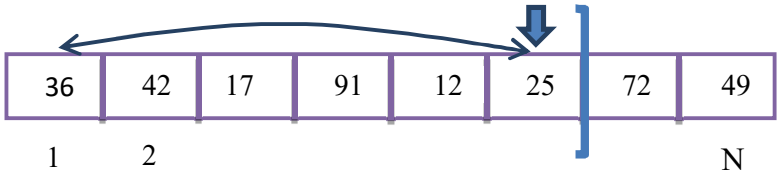


Рисунок 8.2 – Перший обмін елементів масиву

У подальшому потрібно порівнювати опорний елемент із наступним елементом масиву, який стоїть за попереднім місцем розташування опорного елементу (в цьому випадку це елемент *A[N–2]),* і якщо впорядкованість порушується, то необхідно здійснити обмін (рис. 8.3).

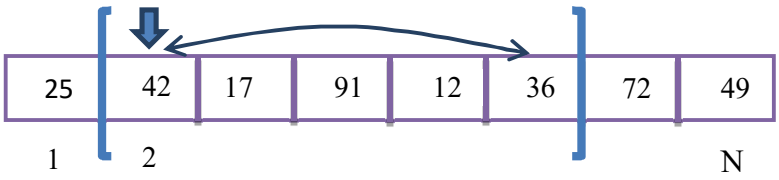


Рисунок 8.3 – Другий обмін елементів масиву

Отже, після кожного обміну потрібно міняти «кінець» масиву, з яким порівнюється значення опорного елемента. При цьому неаналізована частина завжди буде залишатися всередині, зліва від неї будуть елементи, менші за опорний елемент, а справа – більші (рис. 8.4).

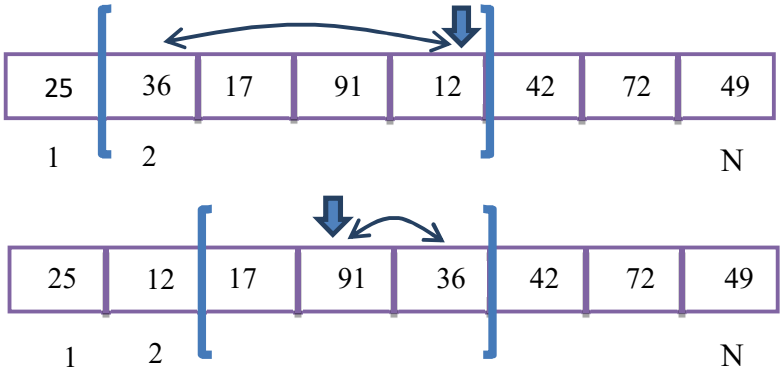


Рисунок 8.4 – Наступні обміни елементів масиву

Після того як опорний елемент порівняно з усіма елементами у масиві, він стане на своє місце, тобто зліва від опорного елемента стануть усі елементи, менші від нього, а справа – більші (рис. 8.5).

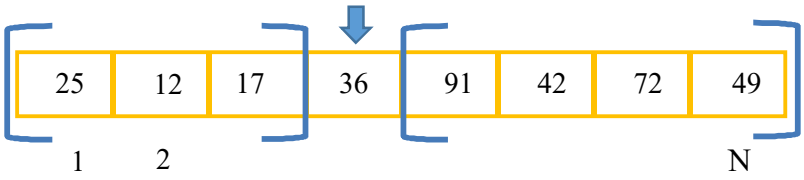


Рисунок 8.5 – Розташування опорного елемента масиву

У подальшому достатньо відсортувати лише праву та ліву частини масиву, а положення опорного елемента вже не мінятиметься.

Швидке сортування реалізує стратегію «поділяй та володарюй» [5]. Процес сортування масиву *A[p, r]* складається з трьох етапів, як і всі алгоритми з використанням декомпозиції.

Етап розділення. Масив *A[p, r]* розбивається на два підмасиви *A[p, q–1]* та *A[q+1, r]*, таких що кожен елемент *A[p, q–1]* менше або дорівнює *A[q]),* який зі свого боку не перевищує будь-який елемент підмасиву *A[q+1, r]*.

Етап володарювання. Підмасиви *A[p, q–1]* та *A[q+1, r]* сортуються за допомогою рекурсивного виклику процедури швидкого сортування.

Етап комбінування. Оскільки підмасиви сортуються на місці, для їх об'єднання не потрібно робити ніякі дії: весь масив *A[p, r]* є отсортованим.

Ідея алгоритму полягає в пошуку «середнього» елемента та перестановці елементів масиву в такий спосіб, щоб його можна було розділити на дві частини й кожний елемент із лівої частини був не більший за «середній» елемент, а елементи з правої частини були більше «середнього». Упорядкування кожної з частин відбувається рекурсивно. Після цього необхідно відсортувати створені частини масиву, і так буде виконуватися доти, доки всі елементи не стануть на свої місця.

З-поміж переваг сортування Хоара можна виокремити:

– один із найбільш швидкодіючих на практиці з алгоритмів внутрішнього сортування загального призначення;

– простий в реалізації;

– добре поєднується з механізмами кешування й віртуальної пам'яті;

– допускає природне розпаралелювання;

– допускає ефективну модифікацію для сортування за декількома ключами;

– працює на структурах із послідовним доступом, що допускають ефективний прохід як від початку до кінця, так і від кінця до початку.

Серед недоліків сортування Хоара можна виокремити:

– пряма реалізація у вигляді функції з двома рекурсивними викликами може привести до помилки переповнення стека, оскільки в гіршому випадку їй може знадобитися зробити *O(n)* вкладених рекурсивних викликів;

– нестійкий алгоритм.

Пірамідальне сортування (*Heapsort*, «сортування купою») ґрунтується на організації елементів у масиві по типу двійкового (бінарного) дерева. *Бінарним деревом* називають ієрархічну структуру даних, у якій кожен елемент має не більше двох нащадків. Піраміда становить особливий вид бінарного дерева, в якому значення кожного елемента більше, ніж значення кожного з його нащадків. Безпосередні нащадки кожного вузла не впорядковані, тому іноді лівий нащадок може виявитися більше правого, а іноді – навпаки. Піраміда становить повне дерево, в якому заповнення нового рівня починається тільки після того, як попередній рівень повністю заповнений, а всі вузли на одному рівні заповнюються послідовно. Бінарна купа – це масив, який можна розглядати як повне бінарне дерево [13–18].

Інакше кажучи, бінарне дерево – це дерево, у якого виконані такі умови:

1) кожен лист має глибину або *d*, або *d–1*, де *d* – максимальна глибина дерева;

2) значення в будь-якій вершині не менші (інший варіант – не більші) за значення їхніх нащадків.

Зручна структура даних для бінарного дерева – такий масив *Array*, що *Array[1*] – елемент у корені, а нащадками *Array[i]* є *Array[2i]* і *Array[2i+1*].

Алгоритм сортування складається з двох основних кроків:

1) представлення елементів масиву у вигляді бінарного дерева:

$$Array[i] > Array[2i]$$
$$Array[i] > Array[2i+1],\ 1 \le i < n/2.$$

Цей крок вимагає *O(n)* операцій;

2) видалення елементів з кореня по одному за раз і перебудова дерева. Тобто на першому кроці виконується обмін елемента *Array[1]* і елемента *Array[n],* перетворення елементів *Array[1], Array[2], ... , Array[n–1]* у бінарне дерево.

У подальшому виконується перестановка *Array[1]* і *Array[n–1]* і перетворення *Array[1], Array[2], ... , Array[n–2]* у бінарне дерево. Процес продовжується доти, доки в бінарному дереві не залишиться один елемент. У такому випадку вважається, що *Array[1], Array[2], ... , Array[n]* – це впорядкована послідовність.

Цей крок вимагає *O(n log n)* операцій [14].

Візуалізація алгоритму пірамідального сортування наведено на рисунку 8.6 [5].

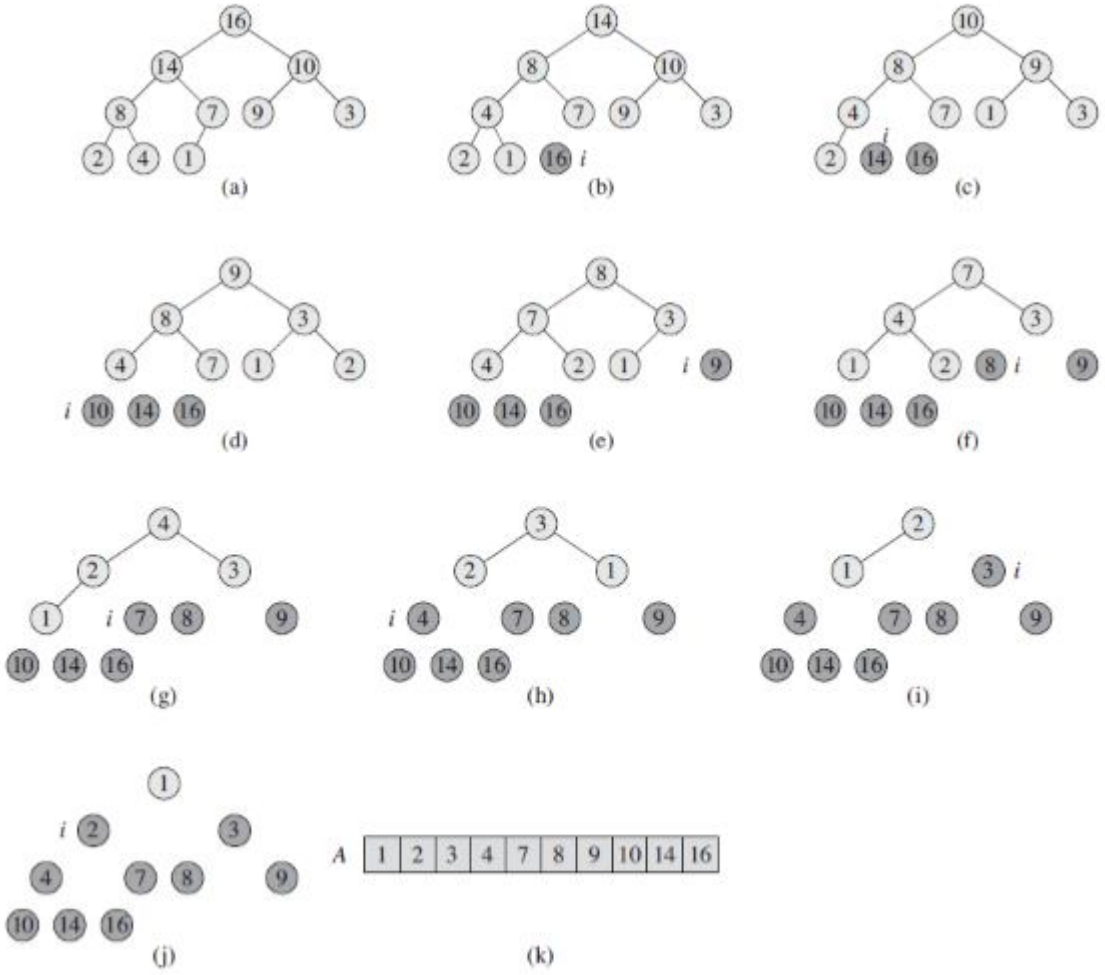


Рисунок 8.6 – *Heap Sort* алгоритм

Пірамідальне сортування працює в будь-якому випадку гарантовано за *O(n log n)* операцій при сортуванні *n* елементів. Кількість застосовуваної службової пам'яті не залежить від розміру масиву, тобто *O(1).*

Просіювання − це побудова нової піраміди за таким алгоритмом: новий елемент поміщається у вершину дерева, далі він переміщається («просівається») по шляху вниз на основі порівняння з дочірніми елементами. Спуск завершується, якщо результат порівняння з дочірніми елементами відповідає ключу сортування.

Загальна ідея пірамідального сортування полягає в тому, що спочатку будується піраміда з елементів вихідного масиву, а потім здійснюється сортування елементів. Виконання алгоритму розбивається на два етапи:

– побудова піраміди. Необхідно визначити праву частину дерева, починаючи з нижнього рівня дерева. Для цього обирається елемент лівіше цієї частини масиву, який просіюється крізь піраміду по шляху, де знаходяться менші його елементи, що одночасно піднімаються вгору. З двох можливих шляхів потрібно обрати шлях через менший елемент. У результаті просіювання найменший елемент виявляється на вершині піраміди;

– сортування на побудованій піраміді. Необхідно обрати останній елемент масиву в якості поточного та поміняти верхній (найменший) елемент масиву й поточний місцями. Поточний елемент, що є в цей час верхнім, просіюється крізь $(n-1)$-елементну піраміду. Після чого опрацьовується передостанній елемент і так виконується аналогічно. У результаті масив буде відсортований за спаданням.

Задача 1. Побудувати блок-схему алгоритму для сортування елементів одномірного масиву, використовуючи швидке сортування.

Розв'язання. Блок-схеми алгоритму для сортування елементів масивів методом швидкого сортування наведено на рисунках 8.7 (варіант 1).

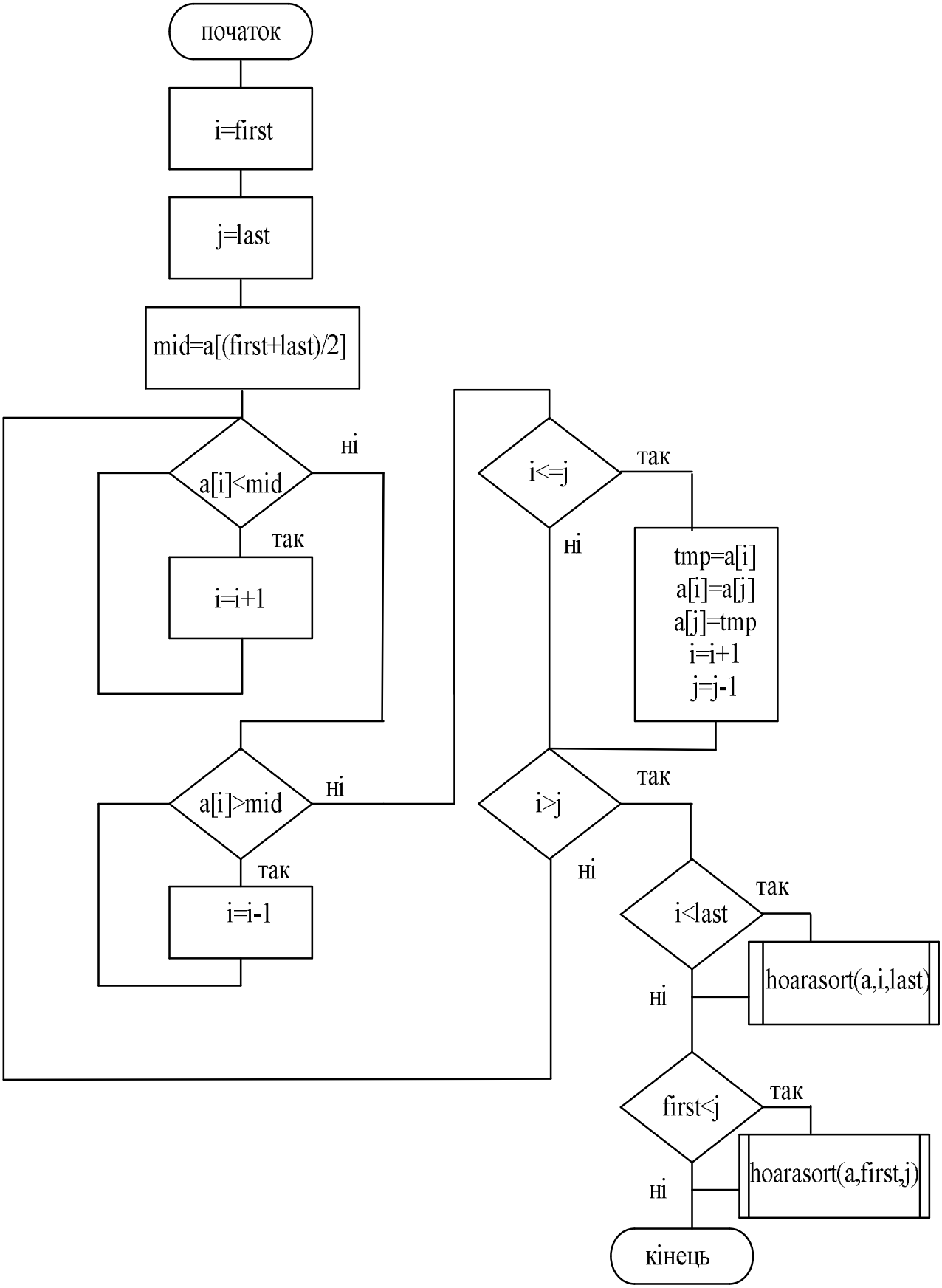


Рисунок 8.7 – Алгоритм швидкого сортування (варіант 1)

Розв'язання. Блок-схему алгоритму для сортування елементів масивів методом швидкого сортування наведено на рисунку 8.8 (варіант 2).

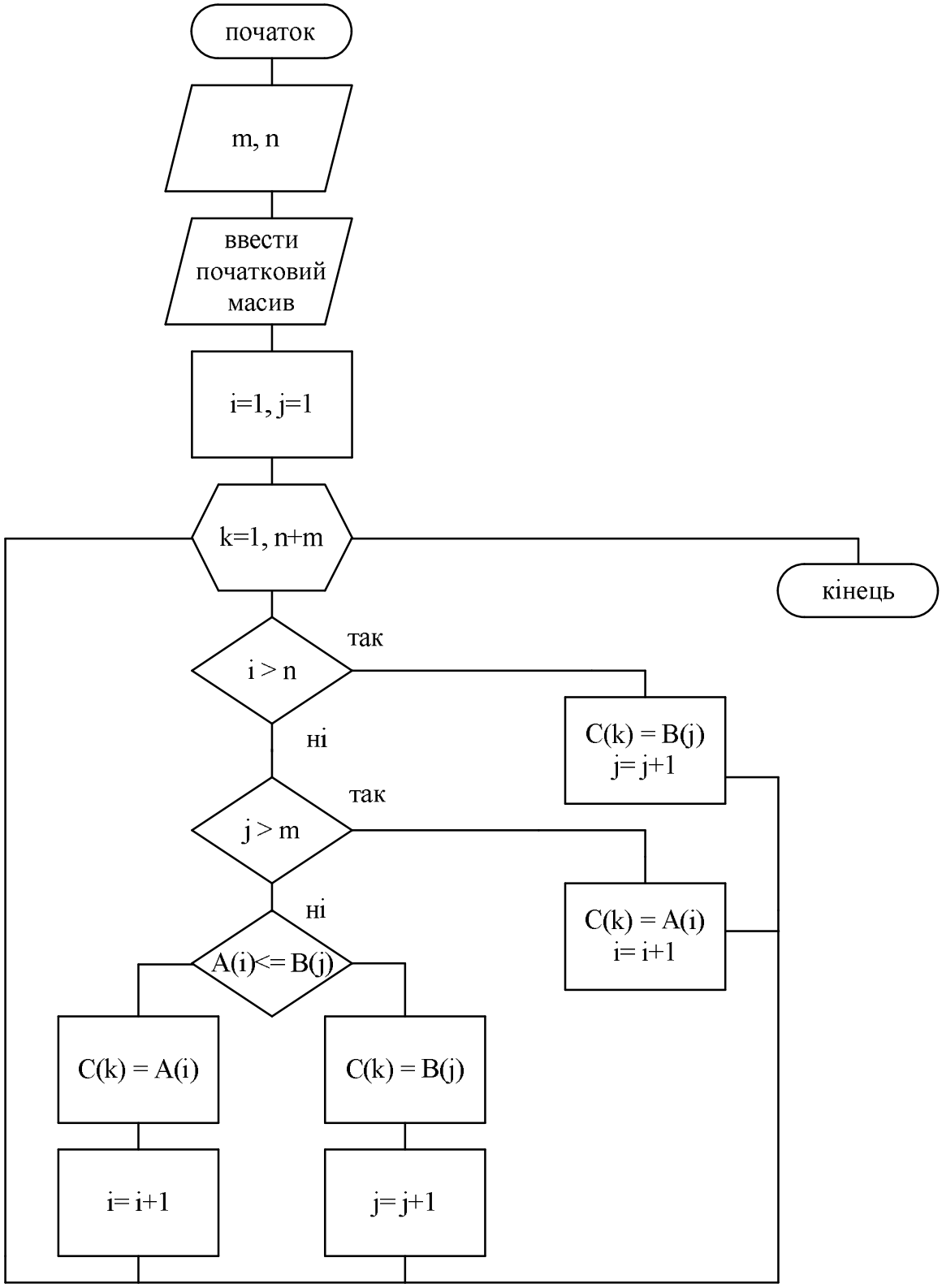


Рисунок 8.8 – Алгоритм швидкого сортування (варіант 2)

Розв'язання. Блок-схему алгоритму для сортування елементів масивів методом швидкого сортування наведено на рисунку 8.9 (варіант 3).

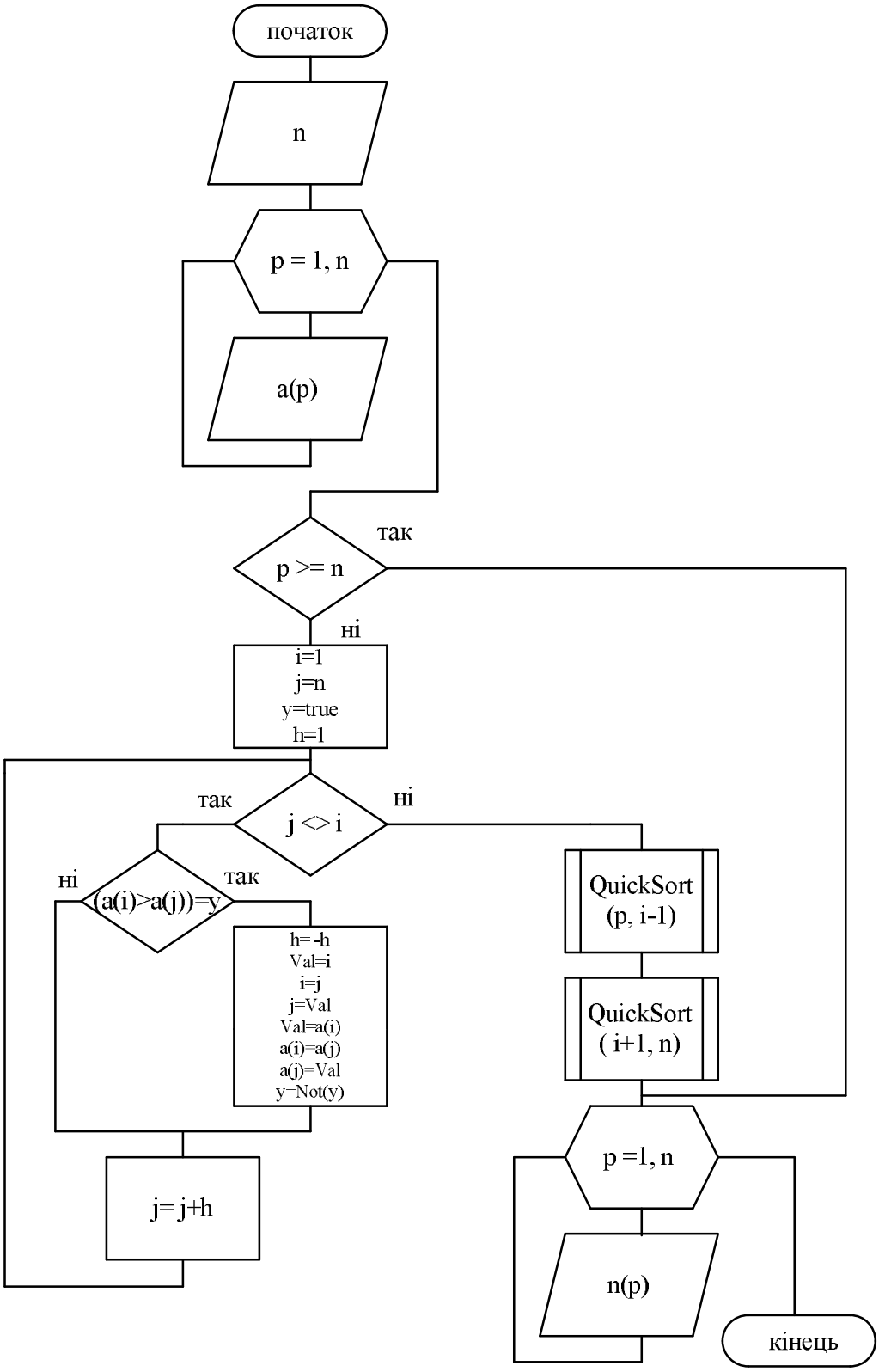


Рисунок 8.9 – Алгоритм швидкого сортування (варіант 3)

Задача 2. Побудувати блок-схему алгоритму для сортування елементів одновимірного масиву, використовуючи пірамідальне сортування.

Розв'язання. Блок-схему алгоритму для сортування елементів масивів пірамідальним сортування наведено на рисунках 8.10 – 8.11 (варіант 1 та варіант 2).

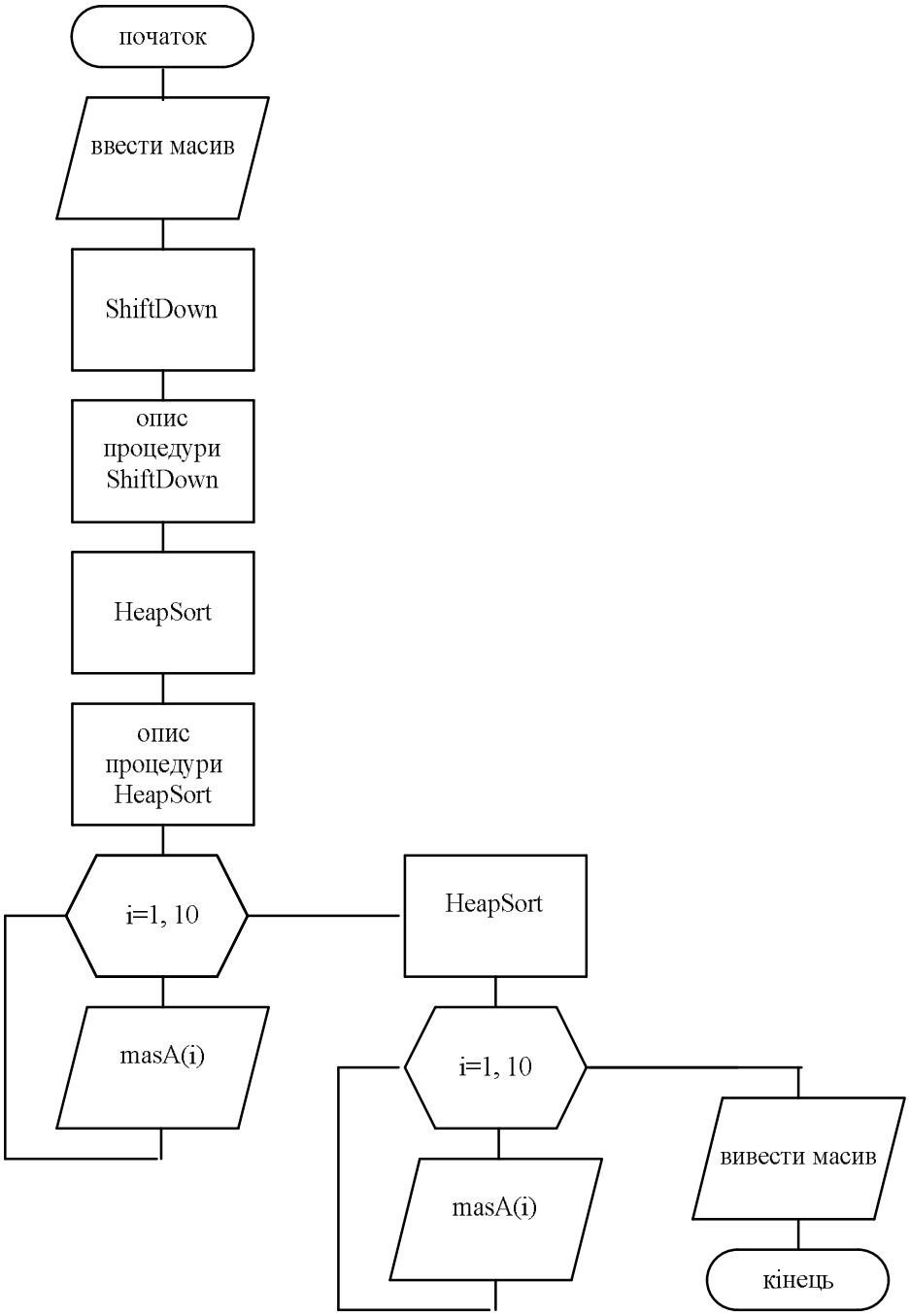


Схема 8.10 – Блок-схема пірамідального сортування (варіант 1)

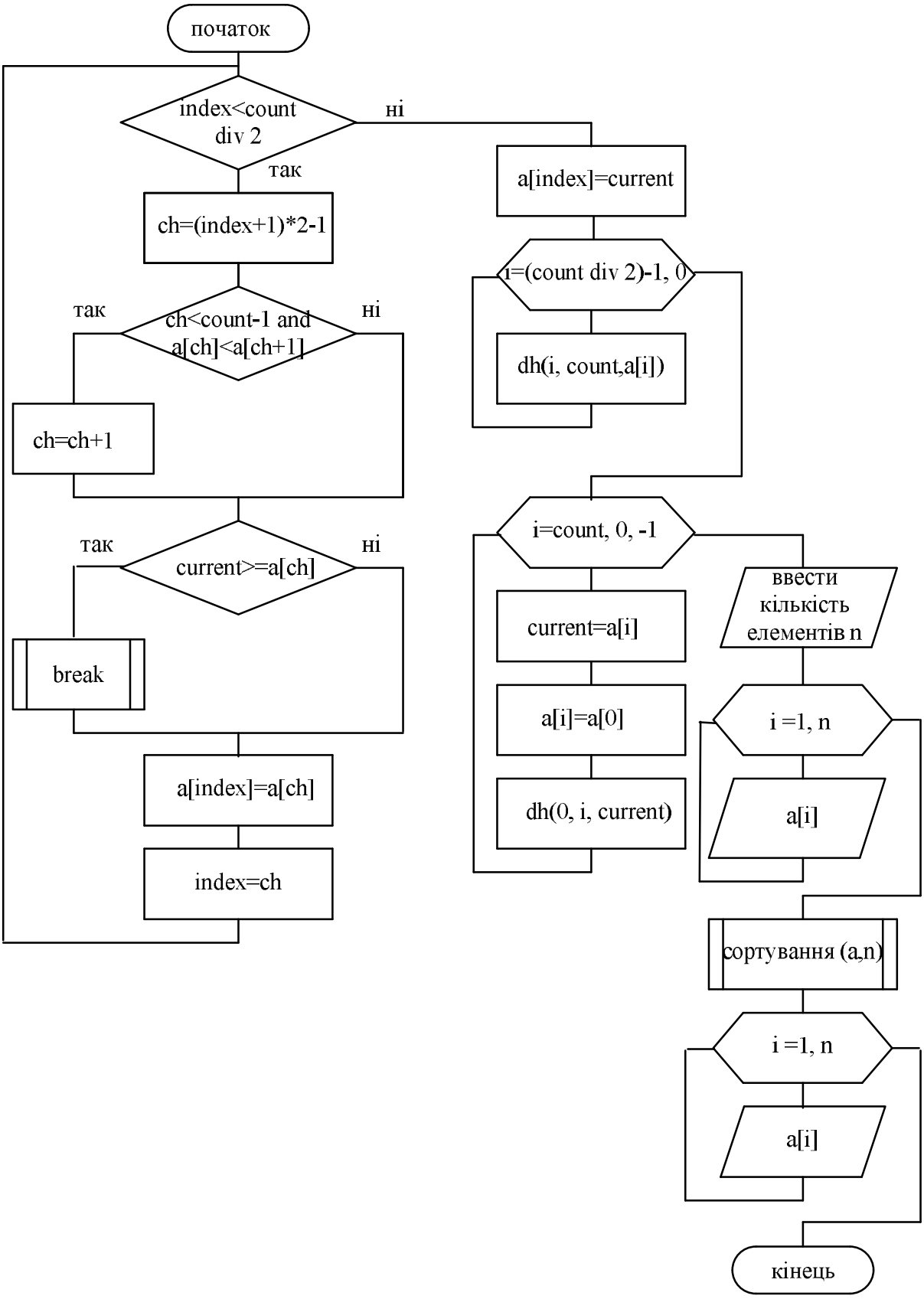


Схема 8.11 – Блок-схема пірамідального сортування (варіант 2)

Для відпрацювання сортування Хоара та пірамідального сортування використати завдання, які наведено в таблиці 5.2.

Контрольні питання

1. Сформулювати мету процесу сортування.
2. Швидке сортування є поліпшеним варіантом якого алгоритму сортування?
3. Яка принципова відмінність сортування Хоара від менш ефективнішого алгоритму, який він вдосконалює?
4. Який елемент обирають як опорний в алгоритмі швидкого сортування? Навести приклад.
5. Яку стратегію реалізує алгоритм Хоара? Охарактеризувати цю стратегію.
6. З яких етапів складається стратегія, реалізована в алгоритмі швидкого сортування?
7. Охарактеризувати переваги та недоліки швидкого сортування
8. Охарактеризувати бінарне дерево.
9. Яким чином визначається бінарне дерево?
10. Надати визначення поняттю «піраміда».
11. З яких кроків складається пірамідальне сортування. Охарактеризувати ці кроки.
12. Яка процедура дає змогу перетворити початкове дерево так, щоб значення вузлів-батьків було більше значення вузлів-дітей?
13. Яка процедура будує дерево з максимальним елементом у корені та з максимальними елементами в наступних рівнях

9 АЛГОРИТМІЗАЦІЯ У ДВОВИМІРНИХ МАСИВАХ

Двовимірний масив трактується як одновимірний масив, тип елементів якого також є масивом (масив масивів). Положення елементів у двовимірних масивах описується двома індексами, які можна подати у вигляді прямокутної таблиці або матриці.

Розглянемо двовимірний масив з розмірністю 3 × 3, тобто в масиві буде три рядки, а в кожному рядку по три елементи:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Кожен елемент двовимірного масиву має свій номер, так само як і кожен елемент одновимірного масиву, але його номер складається з двох чисел: номера рядка, в якому знаходиться елемент, та номера стовпчика. Отже, номер елемента визначається перетином рядка та стовпчика. Наприклад, a_{21} – це елемент, що розташовується в другому рядку й першому стовпчику.

Оголошення *int A[n]* створює в пам'яті одновимірний масив: набір пронумерованих елементів, що йдуть у пам'яті послідовно. До кожного елемента масиву можна звернутися, вказавши один індекс – номер елемента. Але можна створити й двовимірний масив у такий спосіб: *int A[n][m]*. Дане оголошення створює масив із *n* об'єктів, кожен зі свого боку є масивом типу *int [m]*. Тоді *A[i]*, де *i* приймає значення від 0 до *n*–1 буде зі свого боку одним з *n* створених звичайних масивів, і звернутися до елемента з номером *j* у цьому масиві можна записом *A[i][j]* [3; 5; 6].

Приклади розв'язання задач

Задача 1. Побудувати блок-схему алгоритму для формування з двовимірного масиву *A*(6, 9) одновимірного *B*(6), елементами якого є суми кожного рядка масиву *A*, а також визначення максимального елемента масиву *B* та його індексу.

Розв'язання. Блок-схему алгоритму формування з двовимірного масиву *A*(6, 9) одновимірного *B*(6), елементами якого є суми кожного рядка масиву *A*, а також визначення максимального елемента масиву *B* та його індексу наведено на рисунку 9.1.

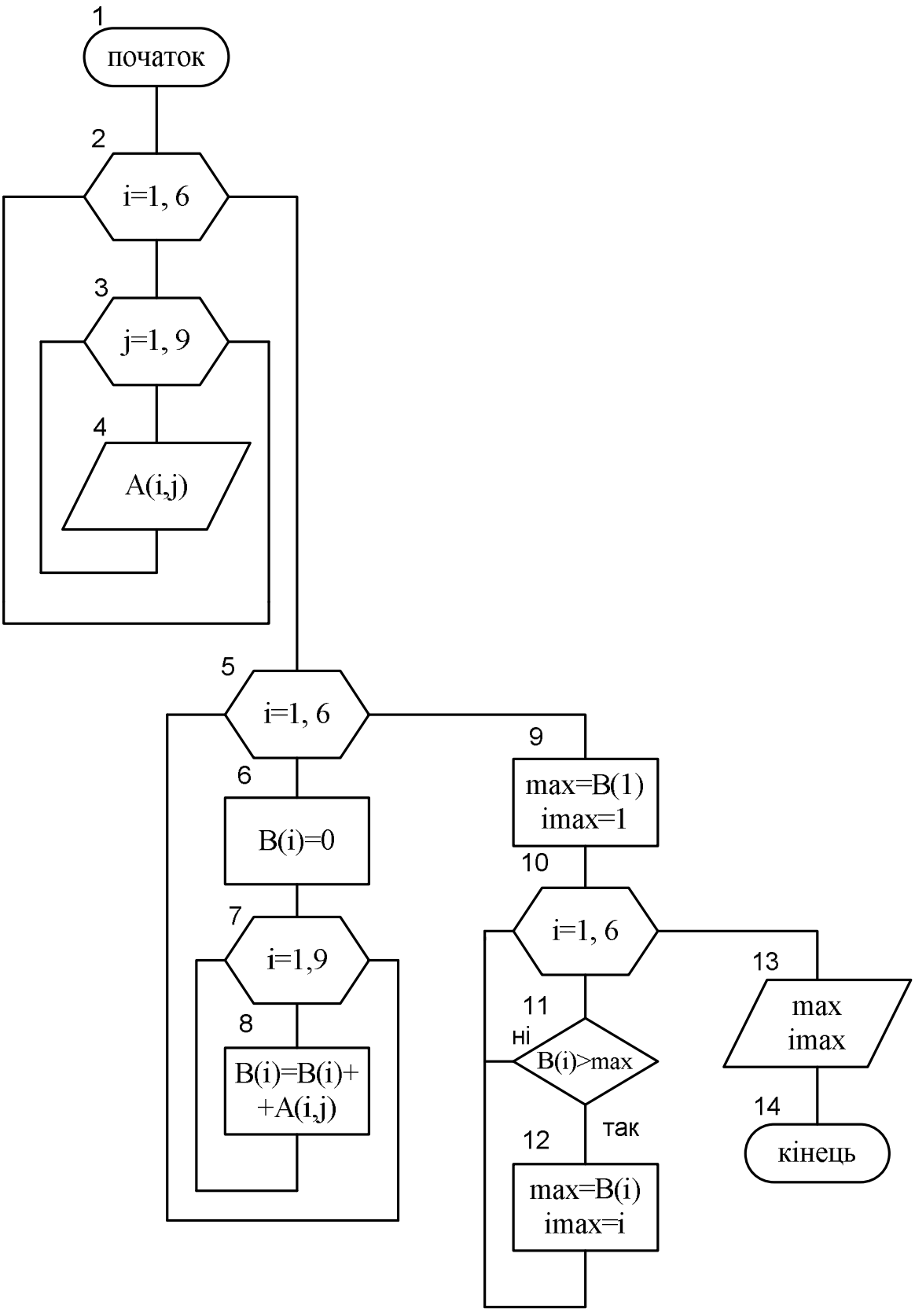


Рисунок 9.1 – Блок-схема алгоритму формування з двовимірного масиву одновимірного

Задача 2. Побудувати блок-схему алгоритму для відшукання середнього арифметичного значення від'ємних елементів у кожному рядку двовимірного масиву.

Розв'язання. Блок-схему алгоритму для відшукання середнього арифметичного значення від'ємних елементів у кожному рядку двовимірного масиву наведено на рисунку 9.2.

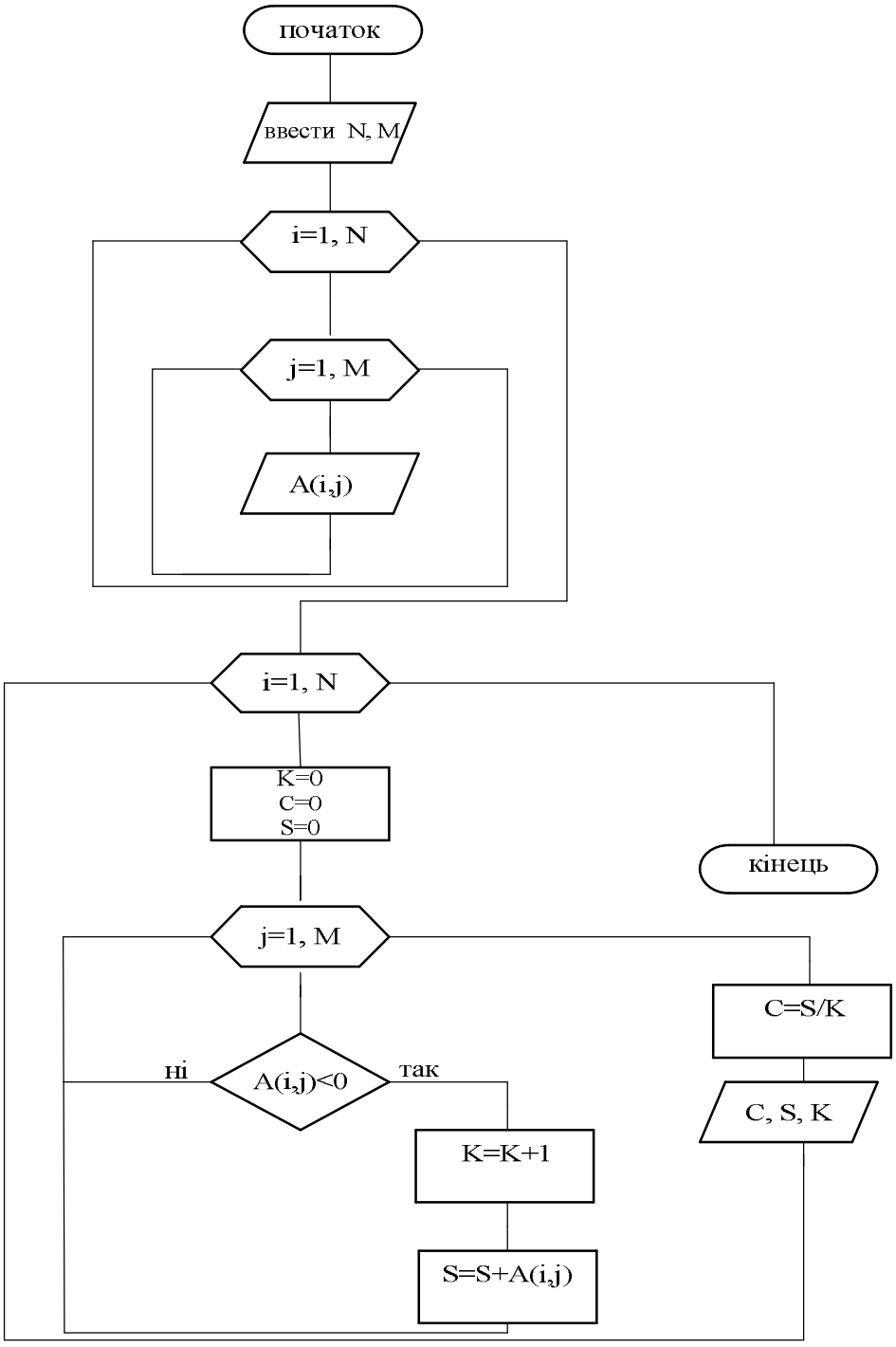


Рисунок 9.2 – Блок-схема алгоритму для відшукання середнього арифметичного значення від'ємних елементів у кожному рядку двовимірного масиву

Задача 3. Побудувати блок-схему алгоритму для відшукання максимального елемента двовимірного масиву.

Розв'язання. Блок-схему алгоритму для відшукання максимального елемента двовимірного масиву (рис 9.3).

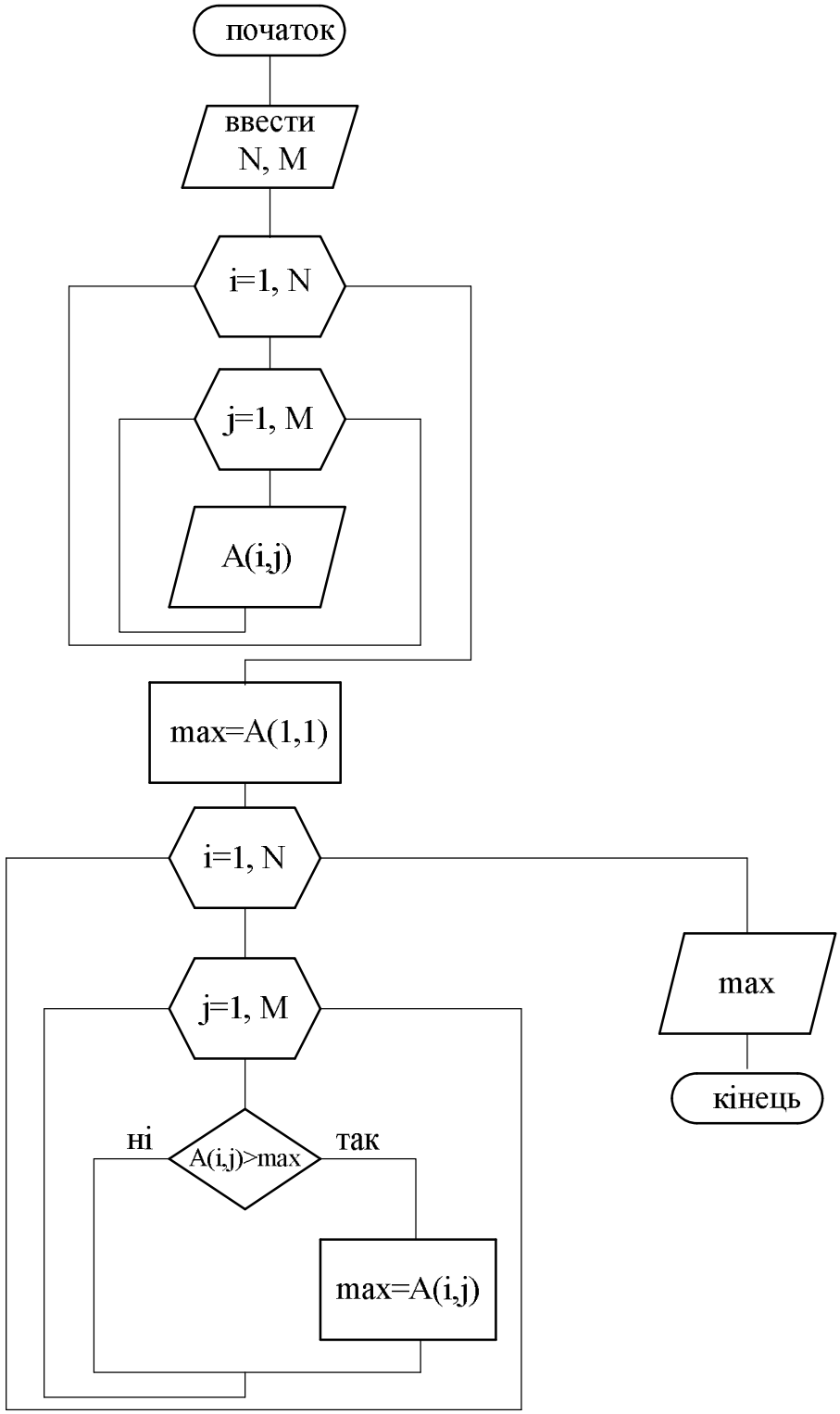


Рисунок 9.3 – Блок-схема алгоритму для відшукання максимального елемента двовимірного масиву

Задача 4. Побудувати блок-схему алгоритму для відшукання максимального елемента на головній діагоналі двовимірного масиву.

Розв'язання. Блок-схему алгоритму для відшукання максимального елемента на головній діагоналі двовимірного масиву наведено на рисунку 9.4.

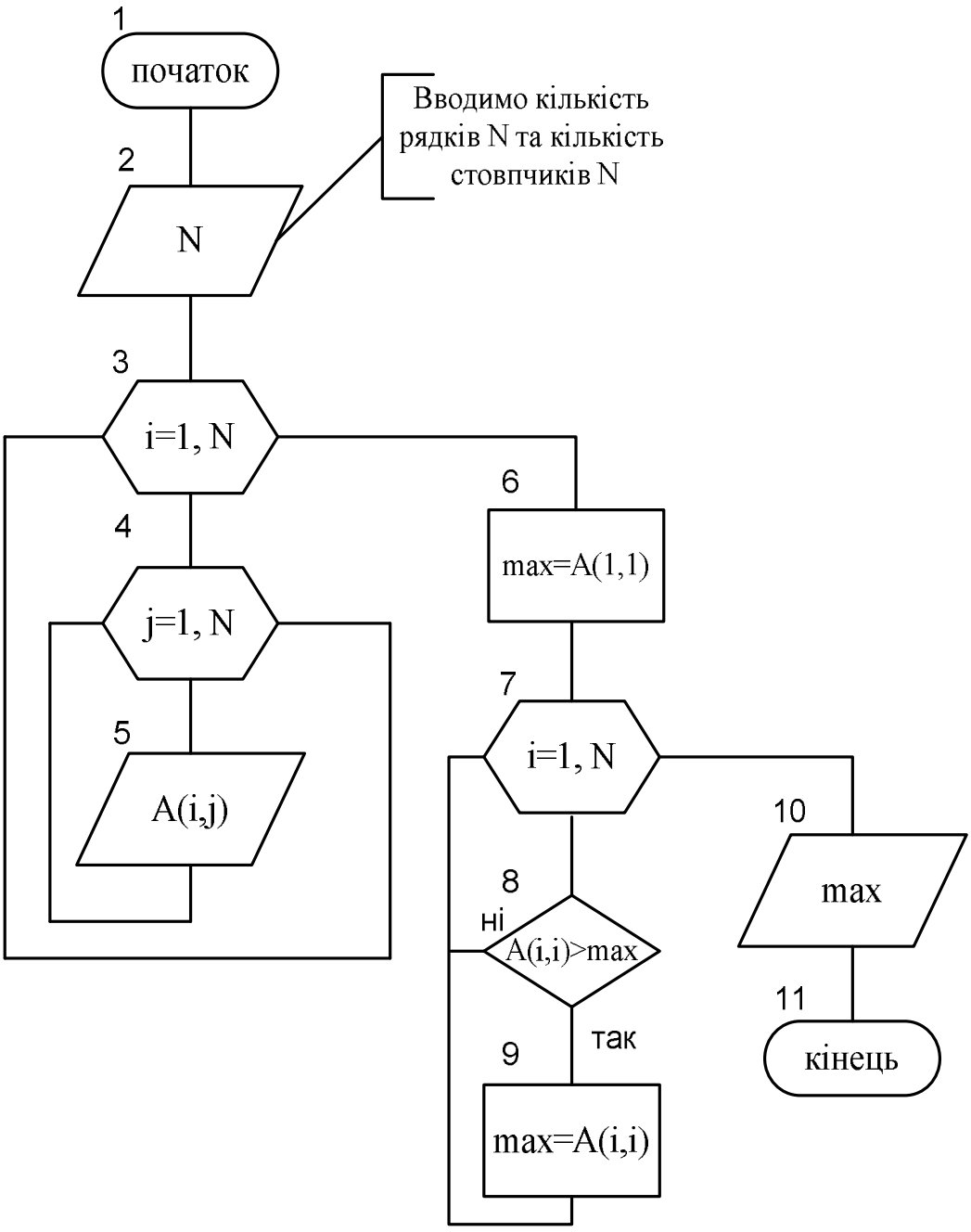


Рисунок 9.4 – Блок-схема алгоритму для відшукання максимального елемента на головній діагоналі двовимірного масиву

Задача 5. Побудувати блок-схему алгоритму для пошуку мінімального елемента нижче головної діагоналі двомірного масиву.

Розв'язання. Блок-схему алгоритму для пошуку мінімального елемента нижче головної діагоналі двовимірного масиву наведено на рисунку 9.5 (варіант 1).

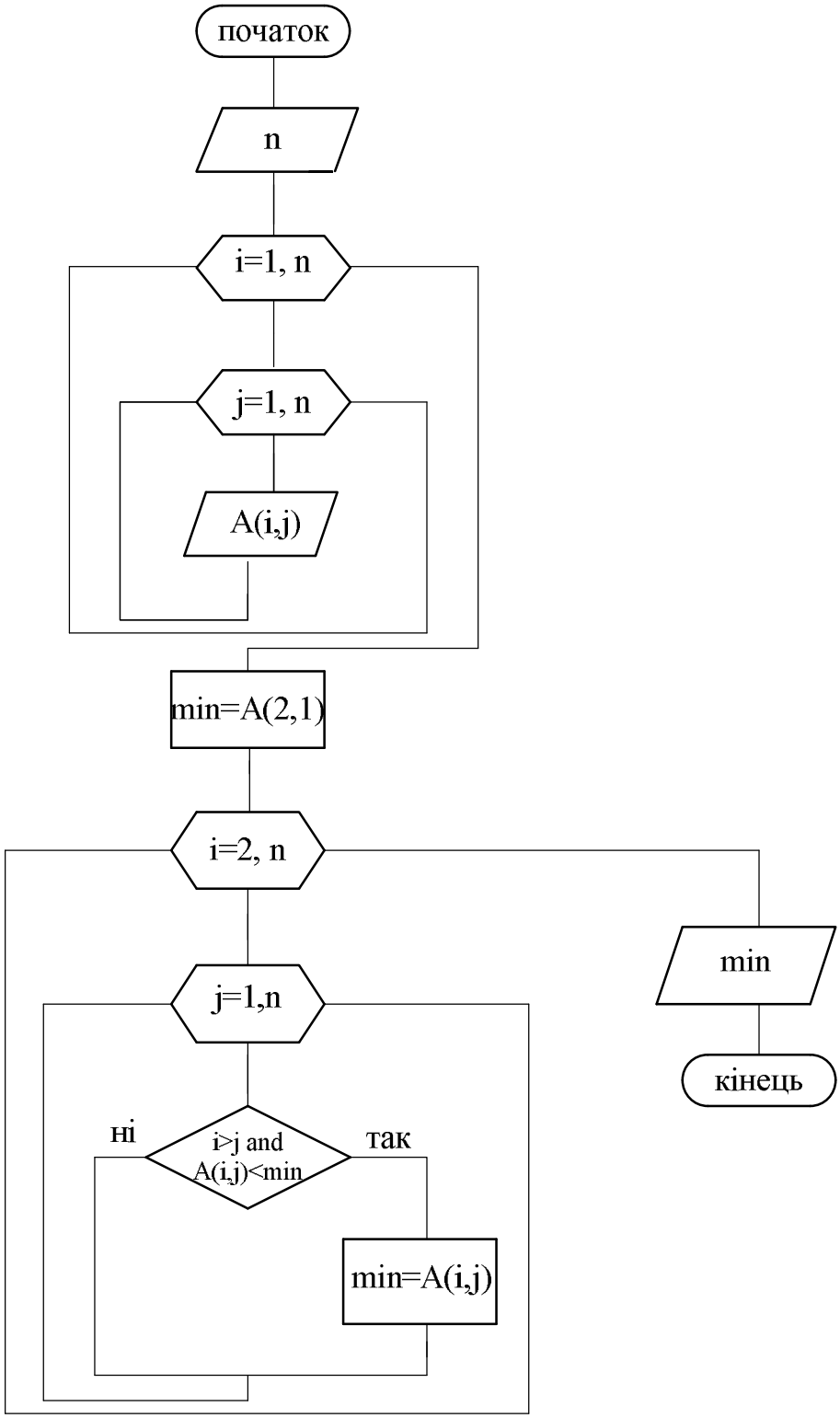


Рисунок 9.5 – Блок-схема алгоритму для відшукання мінімального елемента нижче головної діагоналі двовимірного масиву (варіант 1)

Розв'язання. Блок-схему алгоритму для пошуку мінімального елемента нижче головної діагоналі двовимірного масиву наведено на рисунку 9.6 (варіант 2).

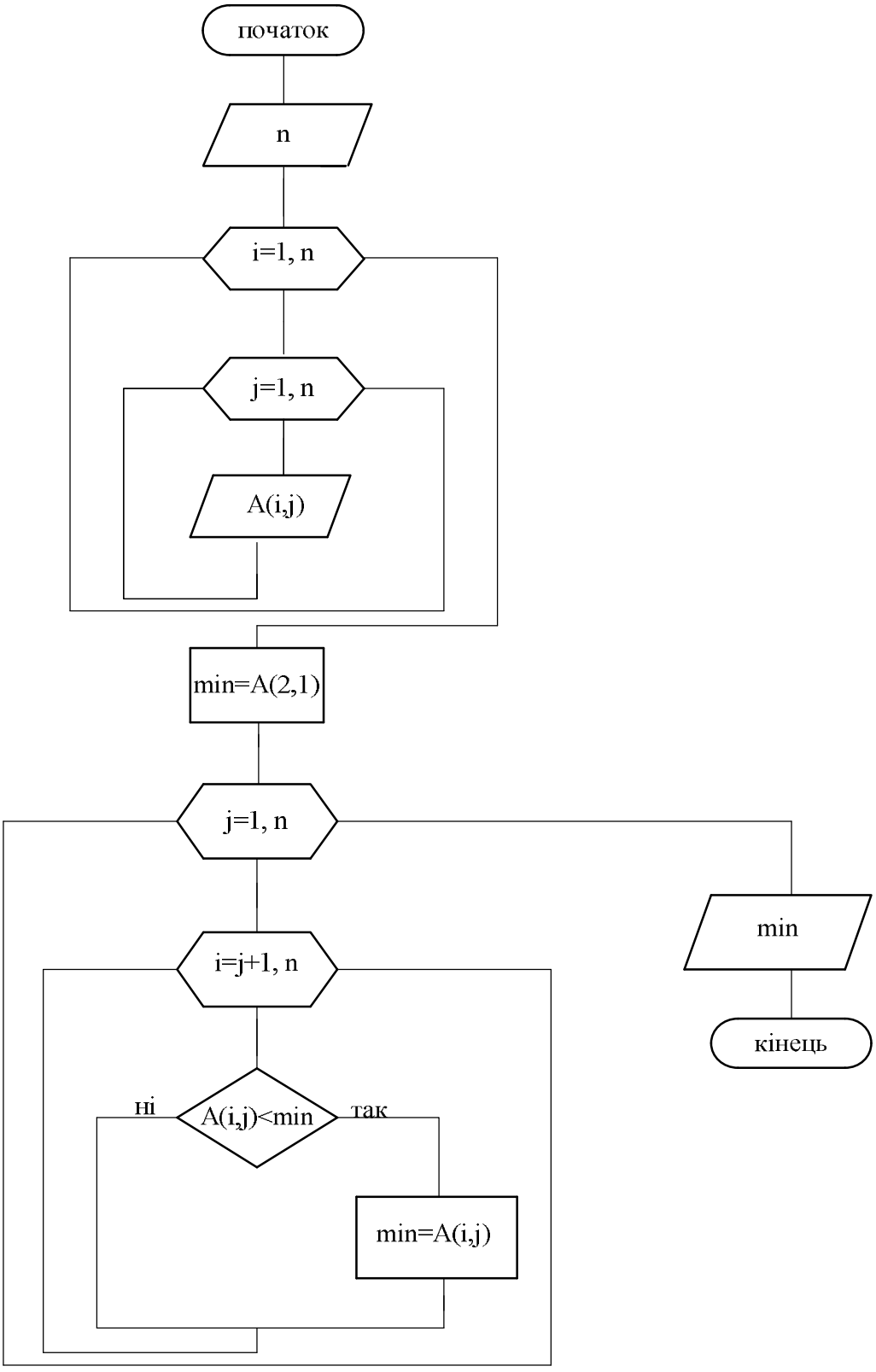


Рисунок 9.6 – Блок-схема алгоритму для відшукання мінімального елемента нижче головної діагоналі двовимірного масиву (варіант 2)

Задача 6. Побудувати блок-схему алгоритму для відшукання мінімального елемента вище головної діагоналі двомірного масиву.

Розв'язання. Блок-схему алгоритму для відшукання мінімального елемента вище головної діагоналі двомірного масиву наведено на рисунку 9.7. (варіант 1).

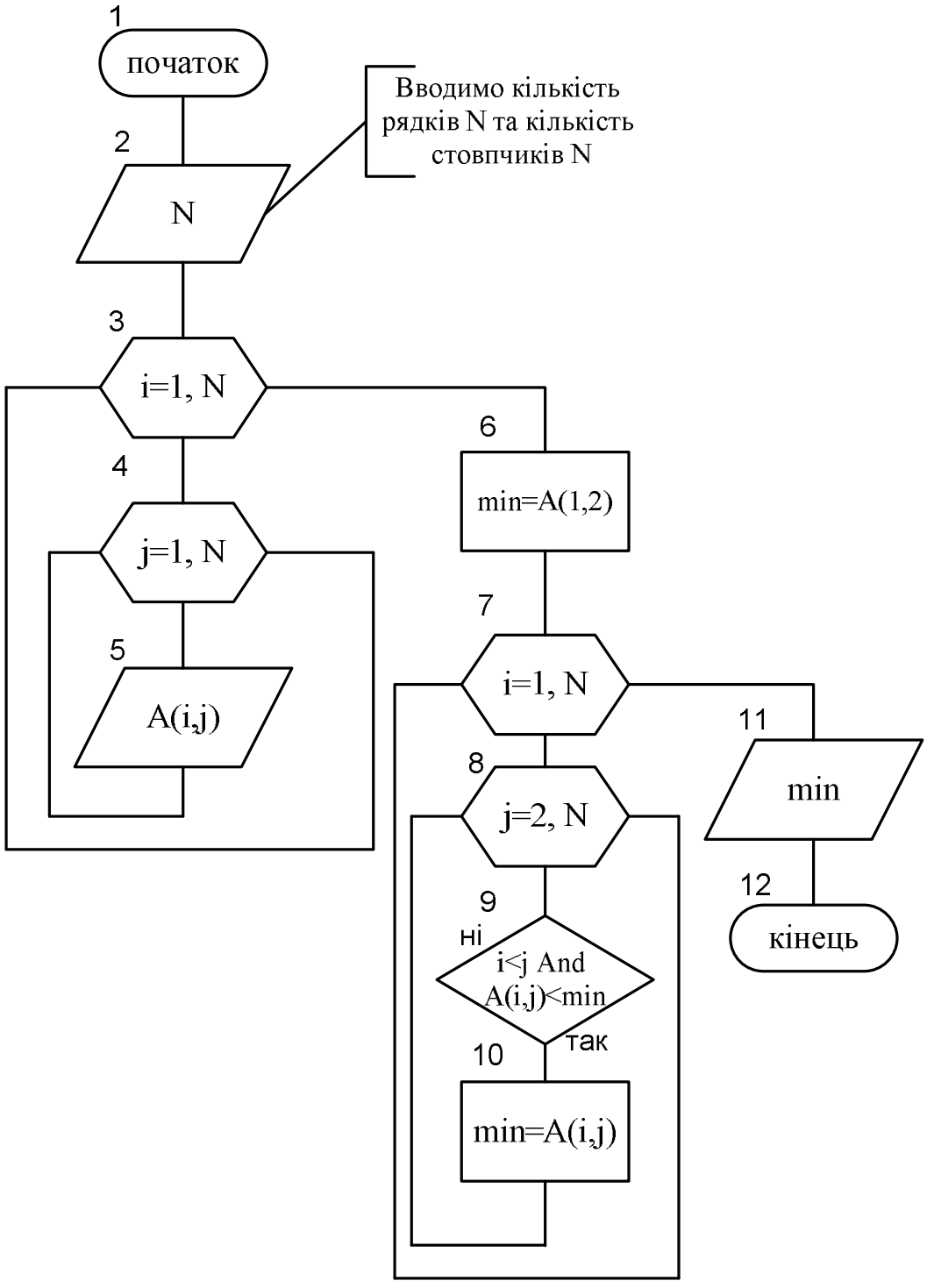


Рисунок 9.7 – Блок-схема алгоритму для відшукання мінімального елемента вище головної діагоналі двомірного масиву (варіант 1)

Розв'язання. Блок-схему алгоритму для відшукання мінімального елемента вище головної діагоналі двомірного масиву наведено на рисунку 9.8 (варіант 2).

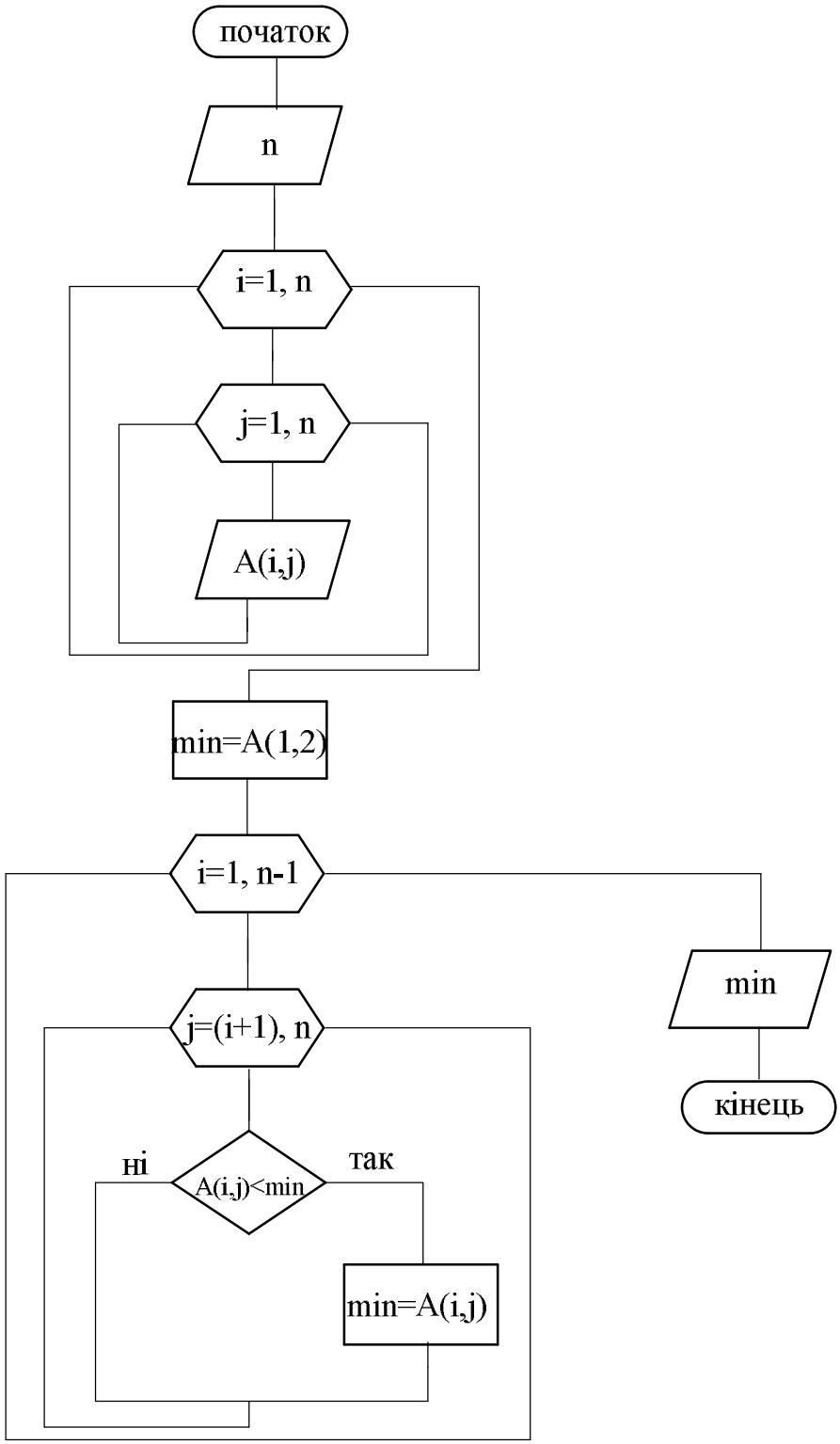


Рисунок 9.8 – Блок-схема алгоритму для відшукання мінімального елемента вище головної діагоналі двомірного масиву (варіант 2)

Задача 7. Побудувати блок-схему алгоритму для відшукання мінімального елемента на побічній діагоналі двовимірного масиву.

Розв'язання. Блок-схему алгоритму для відшукання мінімального елемента на побічній діагоналі двомірного масиву наведено на рисунку 9.9 (варіант1).

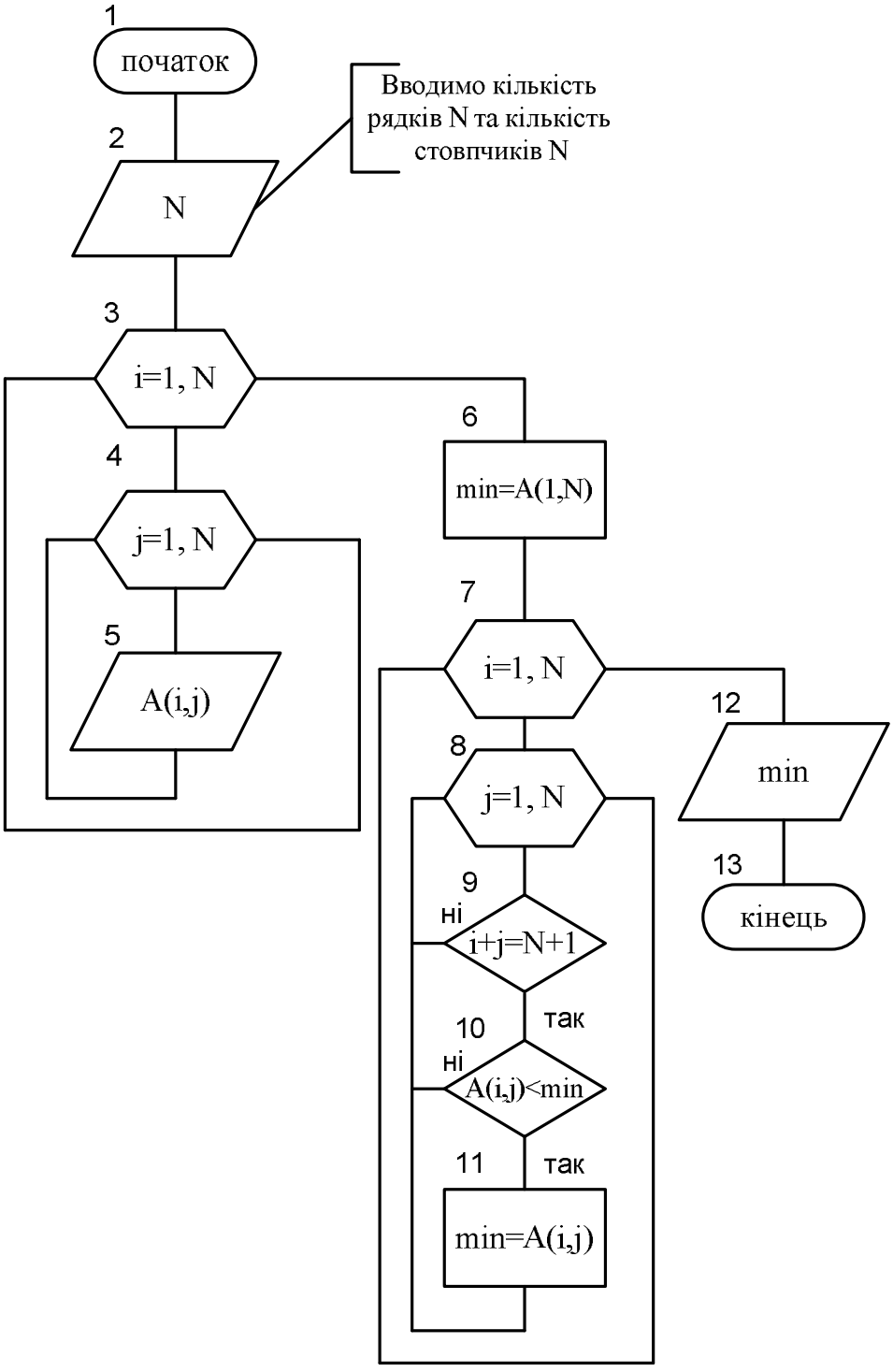


Рисунок 9.9 – Блок-схема алгоритму для відшукання мінімального елемента на побічній діагоналі двовимірного масиву (варіант 1)

Розв'язання. Блок-схему алгоритму для відшукання мінімального елемента на побічній діагоналі двомірного масиву наведено на рисунку 9.10 (варіант 2).

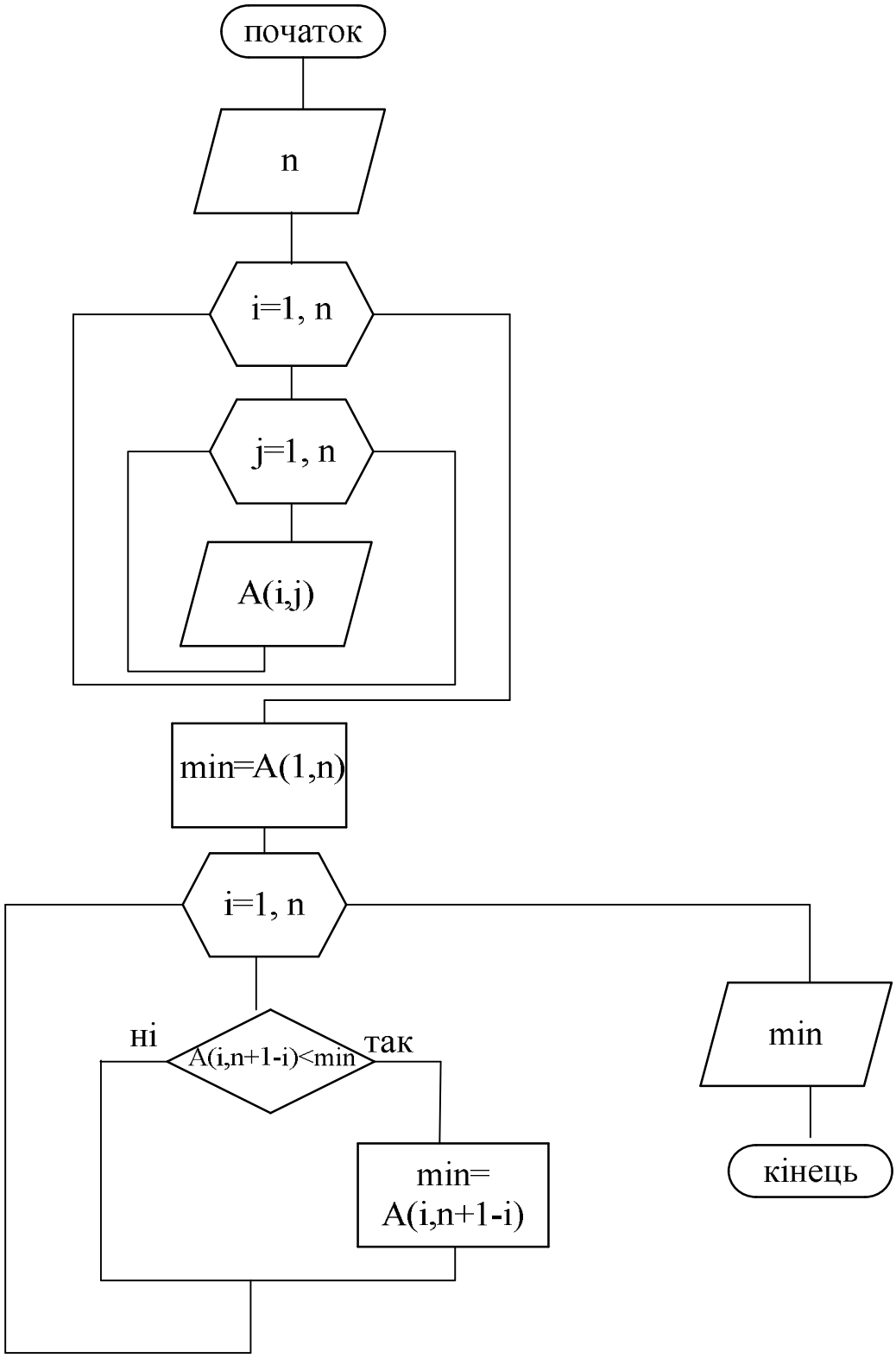


Рисунок 9.10 – Блок-схема алгоритму для відшукання мінімального елемента на побічній діагоналі двовимірного масиву (варіант 2)

У таблиці 9.1 наведені завдання до практичних занять та самостійного вивчення теми «Обробка двовимірних масивів».

Таблиця 9.1 – Варіанти індивідуальних завдань щодо розрахунків у двовимірних масивах

Номер варіанта	Зміст завдання
1	2
1	Визначити додаток від'ємних елементів у кожному стовпчику. Матриця A(3, 4)
2	Обчислити середнє арифметичне значення додатних елементів у кожному рядку з парним номером. Матриця A (3, 4)
3	Визначити кількість та додаток від'ємних елементів у кожному стовпчику. Матриця A (3, 4)
4	Знайти мінімальне значення на боковій діагоналі масиву. Матриця B (4, 4)
5	Обчислити середнє арифметичне значення додатних елементів, розташованих під головною діагоналлю масиву. Матриця B (4,4)
6	Знайти максимальне додатне значення на головній діагоналі масиву. Матриця B (4, 4)
7	Обчислити добуток додатних елементів, розташованих під головною діагоналлю масиву. Матриця B (4, 4)
8	Визначити число елементів, значення яких є меншим, за середнє арифметичне. Матриця A (3, 4)
9	Знайти мінімальне значення на боковій діагоналі масиву. Матриця B (5, 5)
10	Знайти максимальне значення на боковій діагоналі масиву. Матриця B (4, 4)
11	Визначити число елементів кожного рядка, значення яких є більшим, за середнє арифметичне в цьому самому рядку. Матриця C (5, 3)
12	Знайти мінімальне значення на головній діагоналі масиву. Матриця B (5, 5)
13	Обчислити середнє арифметичне значення додатних елементів у кожному рядку. Матриця B (5, 3)
14	Обчислити середнє арифметичне значення від'ємних елементів, розташованих над головною діагоналлю масиву. Матриця B (5, 5)
15	Обчислити середнє арифметичне значення додатних елементів, розташованих над головною діагоналлю масиву. Матриця B (5, 5)
16	Обчислити й надрукувати добуток кожного рядка. Матриця A (5, 3)

Продовження таблиці 9.1

1	2
17	Обчислити й надрукувати додаток кожного рядка. Матриця A (5, 4)
18	Знайти й надрукувати середнє значення кожного рядка. Матриця B (5, 4).
19	Знайти різницю між мінімальним і максимальним значеннями елементів на боковій діагоналі масиву. Матриця B (4, 4).
20	Знайти різницю між мінімальним і максимальним значеннями елементів масиву. Матриця A (5, 4).
21	Підрахувати кількість додатних та від'ємних елементів масиву. Матриця B (4, 5).
22	Обчислити й надрукувати добуток у непарних рядках. Матриця B (5, 4).
23	Обчислити й надрукувати кількість нульових елементів у парних рядках. Матриця B (5, 4).
24	Обчислити й надрукувати кількість ненульових елементів у непарних стовпчиках. Матриця B (5, 4).
25	Знайти різницю між середнім арифметичним значенням додатних елементів у парних рядках та від'ємних елементів – у непарних рядках. Матриця B (5, 5).
26	Сформувати матрицю, яку можна отримати з первинної діленням її від'ємних елементів на додаток додатних. Матриця A (3, 4), матриця C (3, 4).
27	Обчислити та надрукувати елементи матриці C, кожен з яких дорівнює додатку відповідних елементів заданих матриць A та B. Матриця A (3, 4), матриця B (3, 4).
28	Обчислити добуток усіх елементів масиву, виключаючи нульові елементи. Матриця A (5, 3).
29	Обчислити додаток усіх елементів масиву. Матриця A (5, 4).
30	Сформувати матрицю, яку можна отримати з первинної шляхом діленням її елементів на максимальний елемент масиву. Матриця A (4, 5), матриця C (4, 5).

Контрольні питання

1. Дати визначення масиву. Якою є розмірність двовимірного масиву?
2. Яка матриця називається транспонованою до існуючої матриці?
3. Якою є розмірність квадратичної матриці?
4. Які математичні дії можна виконувати з елементами двовимірного масиву? Навести приклади.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Марков А. А. Теория алгоритмов / А. А. Марков, Н. М. Нагорный. – М. : Мир, 1984. – 311 с.
2. Колмогоров А. Н. Теория информации и теория алгоритмов / А. Н. Колмогоров. – М. : Наука, 1987. – 304 с.
3. Вирт Н. Алгоритмы и структуры данных / Н. Вирт. – М. : Мир, 1989. – 360 с.
4. Ахо А. Структуры данных и алгоритмы / А. Ахо, Дж. Хопкрофт, Дж. Ульман. – М. : Издательский дом «Вильямс», 2001. – 384 с.
5. Алгоритмы : построение и анализ / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. – 2-е изд. – М. : Вильямс , 2005. – 1296 с.
6. Седжвик Р. Фундаментальные алгоритмы на С++. Анализ / Структуры данных / Сортировка / Поиск / Р. Седжвик. – Киев : ДиаСофт, 2001. – 688 с.
7. Мальцев А. И. Алгоритмы и рекурсивне функции / А. И. Мальцев. – М. : Мир, 1965. – 276 с.
8. Трахтенброт Б. А. Алгоритмы и вычислительные автоматы / Б. А. Трахтенброт. – М. : Сов. радио, 1974. – 245 с.
9. Віківерситет. Теорія алгоритмів [Електронний ресурс]. – Режим доступу : https://ru.wikiversity.org/wiki/Теория_алгоритмов.
10. Сортировка Хоара (быстрая сортировка) [Електронний ресурс]. – Режим доступу : http://www.itmathrepetitor.ru/sortirovka-khoara-bystraya-sortirovka/.
11. Швидке сортування [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Швидке_сортування.
12. Швидке сортування – один з кращіх методів сортування масивів [Електронний ресурс]. – Режим доступу до ресурсу: http://stylezhinki.ru/tehnika-ta-tehnologii/11531-shvidke-sortuvannja-odin-z-krashhih-metodiv.html.
13. Tproger. Візуалізації алгоритмів сортування [Електронний ресурс]. – Режим доступу : https://tproger.ru/digest/sorting-algorithms-visualized/.
14. Записки програміста. Пірамідальне сортування [Електронний ресурс]. – Режим доступу : http://axis.bplaced.net/news/98.
15. Пирамидальная сортировка [Електронний ресурс]. – Режим доступу : http://algolist.manual.ru/sort/pyramid_sort.php.
16. Пирамидальная сортировка [Електронний ресурс]. – Режим доступу : https://prog-cpp.ru/sort-pyramid/.
17. Пирамидальная сортировка [Електронний ресурс]. – Режим доступу : https://prog-cpp.ru/sort-pyramid/.
18. Пирамидальная сортировка [Електронний ресурс]. – Режим доступу : http://www.algolib.narod.ru/Sort/Piramidal.html.

Навчальне видання

ПЕТРОВА Олена Олександрівна,
СОЛОДОВНИК Ганна Валеріївна

АЛГОРИТМІЧНІ ЗАДАЧІ ТА ЇХ ВИРІШЕННЯ

НАВЧАЛЬНИЙ ПОСІБНИК

Відповідальний за випуск *М. В. Новожилова*

Редактор *В. І. Шалда*
Комп'ютерне верстання *І. В. Волосожарова*
Дизайн обкладинки *Т. А. Лазуренко*

Підп. до друку 03.04.2020. Формат 60 х 84/16.
Друк на ризографі. Ум. друк. арк. 6,1
Тираж 50 прим. Зам. №

Видавець і виготовлювач:
Харківський національний університет
міського господарства імені О.М. Бекетова,
вул. Маршала Бажанова, 17, Харків, 61002.
Електронний адрес: rectorat@kname.edu.ua
Свідоцтво суб'єкта видавничої справи:
ДК № 5328 від 11.04.2017.