МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

Лінгур Любов Миколаївна

КОНСПЕКТ ЛЕКЦІЙ
по дисципліні
«ІНФОРМАТИКА»

для здобувачів першого бакалаврського рівня освіти
всіх форм навчання
спеціальності 051 Економіка
освітні програми
Економіка підприємства; Економічна кібернетика

№4733-РС-2025

Затверджено на засіданні кафедри
Економічної кібернетики
та інформаційних технологій
Протокол № 1
від 30 « серпня » 2024р.

Одеса – 2024

Автор: к.е.н., доцент Лінгур Любов Миколаївна

Конспект лекцій «Інформатика» призначений для здобувачів першого бакалаврського рівня освіти всіх форм навчання спеціальності 051 Економіка; освітні програми Економіка підприємства; Економічна кібернетика. – Одеса, 128 с.

Резюме.

Конспект лекцій з Інформатики розроблений для студентів першого курсу бакалаврату спеціальності 051 Економіка всіх форм навчання, які прагнуть опанувати фундаментальні концепції, методології та технології створення алгоритмів. Курс закладає міцну основу для проектування та розробки алгоритмів, методів обчислень та програмного забезпечення, використовуючи сучасні засоби. Особливий акцент робиться на набутті практичних навичок розробки програмного забезпечення для вирішення економіко-математичних задач.

Мета вивчення дисципліни: забезпечити систематизоване засвоєння концепції, методології та технології створення алгоритмів.

Завданнями навчальної дисципліни є: формування знань, необхідних для сучасного проектування та розробки алгоритмів, методів обчислень, а також програмного забезпечення; використання сучасних засобів для розбудови алгоритмів та набуття практичних навичок розробки програм для вирішення економіко-математичних задач.

Курс розпочинається з ознайомлення з поняттями задачі, алгоритму, даних та етапами розробки алгоритмів. Далі вивчаються способи представлення алгоритмів, структура програмного забезпечення ПК, операційні та файлові системи. Значна увага приділяється базовим алгоритмічним конструкціям, рекурсії, а також циклічним структурам. Важливою частиною курсу є розробка алгоритмічних структур засобами електронних таблиць, підготовка та аналіз даних через форматування, вивчення баз даних та роботи з ними в електронних таблицях. Завершується курс вивченням проміжних підсумків, консолідації даних, зведених таблиць та візуалізації рішень за допомогою дашбордів.

Практичні заняття дозволять застосувати теоретичні знання. Починаючи з найпростіших алгоритмів та обробки цілих чисел, засвоєння MS Excel для виконання алгоритмів за допомогою формул. Розробка алгоритмів з розгалуженнями, проміжний аналіз даних та консолідацією даних. Окремі заняття присвячені технологіям створення зведених таблиць, роботі з базами даних в MS Excel (включно з розробкою калькулятора страхового відшкодування) та аналізу даних через створення зведених таблиць та дашбордів.

Цей курс призначене забезпечити здобувачів комплексними знаннями та практичними навичками, необхідними для ефективної роботи з

алгоритмами та даними, що стане міцною основою для майбутньої професійної діяльності.

ЗМІСТ

Лекція 1. Поняття задачі. Аналіз процесу розв'язання задачі. Поняття алгоритму, даних та аналізу алгоритму. Основні етапи розробки алгоритмів

1.1 Поняття задачі. Аналіз процесу розв'язання задачі.

«Програма = Алгоритм + Структура даних»

Ця формула дуже точно відображає той факт, що для розробки ефективної програми важливим являється як вибір (або розробка) ефективного алгоритму, так і вибір структури даних, до яких такий алгоритм буде застосовано. Кожна із таких складових частин програми не може бути виключена, оскільки правильний підбір структур даних є надзвичайно важливим для ефективного функціонування відповідних алгоритмів, а вибір ефективних алгоритмів у свою чергу дозволяє оптимізувати використання машинного часу та пам'яті.

Слово «алгоритм» походить від імені узбецького вченого аль Хорезмі, який у IX ст. розробив правила виконання чотирьох арифметичних дій над числами в десятковій системі числення. Сукупність цих правил у Європі почали називати «алгоризм». Пізніше цей термін перетворився у «алгоритм» і ним називали правила розв'язування різних задач.

Першим алгоритмом, що дійшов до нас на інтуїтивному розумінні як скінченна послідовність елементарних дій, які розв'язують поставлену задачу, вважають запропонований Евклідом у III ст. алгоритм знаходження найбільшого спільного дільника 2 чисел.

Аж до 30-х рр XX ст поняття алгоритму мало швидше методологічне значення. Під алгоритмом розуміли скінченну сукупність точно сформульованих правил, які дають змогу розв'язувати задачі певного класу. Таке визначення алгоритму не є строгим, а скоріше інтуїтивним, оскільки спирається на інтуїтивні поняття (правило, клас задач). У 20-х рр XX ст задача строгого визначення алгоритму стала однією із центральних математичних проблем.

Перші фундаментальні праці з теорії алгоритмів опубліковані в середині 30-х років XX ст. Аланом Тьюрінгом, Алоїзом Черчем, Емілем Постом. Запропоновані ними машини Тьюрінга, Поста і клас рекурсивних функцій Черча стали першими формальними описами алгоритму, які використовувалися як строго визначені моделі обчислень. Сформульовані гіпотези Черча-Тьюрінга та Поста постулювали еквівалентність запропонованих ними моделей обчислень як формальних систем та інтуїтивного поняття алгоритму.

1.2 Поняття алгоритму, даних та аналізу алгоритму.

Алгоритм – це формально описана обчислювальна процедура, що отримує вхідні дані, які називають також входом алгоритму, або його аргументом, і вдає результат обчислень на вихід.

Наведемо інші означення алгоритму. Томас Кормен означає алгоритм як формально описану обчислювальну процедуру, яка отримує вхідні дані (їх ще називають входом алгоритму або його аргументом) та повертає на виході результат обчислень. Формулювання задачі описує, яким вимогам повинне задовольняти розв'язання, а алгоритм знаходить об'єкт, який цим вимогам задовольняє.

Під алгоритмом будемо розуміти скінченну чітко сформульовану сукупність вказівок, які визначають послідовність дій виконавця, спрямованих на досягнення мети або розв'язання задач певного класу за скінченний проміжок часу.

Алгоритм можна також розглядати як інструмент, який призначений для вирішення коректно поставленої обчислювальної задачі. У постановці задачі в загальних рисах визначаються відношення між входом та виходом. В алгоритмі описується конкретна обчислювальна процедура, за допомого якої можна досягнути виконання вказаних відношень.

Можна навести загальні риси алгоритму:

1. Дискретність інформації. Кожний алгоритм має справу з даними: вхідними, проміжними, вихідними. Ці дані представляються у вигляді скінченних слів деякого алфавіту.

2. Дискретність роботи алгоритму. Алгоритм виконується по кроках та при цьому на кожному кроці виконується тільки одна операція.

3. Детермінованість алгоритму. Система величин, які отримуються в кожний (не початковий) момент часу, однозначно визначається системою величин, які були отримані в попередні моменти часу.

4. Елементарність кроків алгоритму. Закон отримання наступної системи величин з попередньої повинен бути простим та локальним.

5. Виконуваність операцій. В алгоритмі не повинно бути невиконуваних операцій. Наприклад, не можна в алгоритмі призначити значення змінній «нескінченність», така операція була би невиконуваною. Кожна операція опрацьовує певну ділянку у слові, яке обробляється.

6. Скінченність алгоритму. Опис алгоритму повинен бути скінченним.

7. Спрямованість алгоритму. Якщо спосіб отримання наступної величини з деякої заданої величини не дає результату, то має бути вказано, що це треба вважати результатом алгоритму.

8. Масовість алгоритму. Початкова система величин може обиратись з деякої потенційно нескінченної множини.

Природно, виникає наступне питання: «Для чого вивчати алгоритми?» По-перше, алгоритми є необхідними складовими для рішення будь-яких задач з різноманітних напрямків економіки, фінансів, менеджменту, комп'ютерних наук. Алгоритми відіграють ключову роль у сучасному розвитку технологій. Тут можна згадати такі розповсюджені задачі, як:

- розв'язання математичних рівнянь різної складності;

- знаходження оптимальних шляхів транспортування товарів та людей;
- знаходження оптимальних варіантів розподілення ресурсів між різними вузлами (виробниками, верстатами, працівниками, процесорами тощо);
- знаходження в геномі послідовностей, які співпадають;
- пошук інформації в глобальній мережі Інтернет;
- прийняття фінансових рішень в електронній комерції;
- обробка та аналіз аудіо та відео інформації.

Цей список можна продовжувати й продовжувати і, власне кажучи, майже неможливо знайти таку галузь господарства, де б не використовувались ті або інші алгоритми.

По-друге, якісні та ефективні алгоритми можуть бути каталізаторами проривів у галузях, які є на перший погляд далекими від комп'ютерних наук (квантова механіка, економіка та фінанси, теорія еволюції).

І, по-третє, вивчення алгоритмів це також цікавий процес, який розвиває математичні здібності та логічне мислення.

Теоретичні дослідження та практика виконання алгоритмів підтверджують важливість наступних моментів:

Єдиного методу розробки алгоритмів для розв'язання задач не існує.

1. Існують задачі, для яких алгоритму розв'язання не існує. Це так звані алгоритмічно-нерозв'язні проблеми.
2. Існують важко розв'язувані задачі (алгоритм розв'язування задачі існує, але практично реалізований не може бути).
3. Для розв'язування однієї і тієї ж задачі може існувати декілька еквівалентних алгоритмів, тому потрібно мати методи деякого ефективного відбору алгоритмів.
4. В основу алгоритмів можуть бути покладені різні принципи, що може суттєво вплинути на час розв'язання задачі.
5. Один і той же алгоритм може бути поданий різними способами.
6. Ефективність алгоритму суттєво залежить від способу організації вхідних та вихідних даних.
7. Необхідно чітко вказувати діапазон вхідних та вихідних даних, які обробляються за допомогою алгоритму.
8. Кожен крок алгоритму повинен бути чітко та однозначно сформульований.

1.3 Основні етапи розробки алгоритмів

Розглянемо етапи повної побудови алгоритму.

1. Постановка задачі.
2. Побудова моделі.
3. Розробка алгоритму.
4. Перевірка правильності алгоритму.
5. Реалізація алгоритму.

6. Аналіз алгоритму та його складність.

7. Перевірка програми.

8. Створення документації.

Перш за все розглянемо призначення кожного етапу та з'ясуємо, як ці етапи об'єднуються в одне ціле.

1. Постановка задачі.

Для того, аби почати процес розв'язання задачі, необхідно попередньо її точно сформулювати.

Зазвичай процес формулювання задачі зводиться до постановки правильних питань. Перерахуємо деякі питання, які будуть корисними у процесі формулювання задачі:

- Чи є зрозумілою термінологія, що використовується у попередньому формулюванні?
- Що дано?
- Що потрібно знайти?
- Як визначити розв'язання?
- Яких даних не вистачає і чи усі дані є потрібними?
- Чи є дані, що не використовуються у задачі?
- Які зроблено припущення?

Є можливими й інші питання в залежності від конкретної задачі. Часто для отримання повних або часткових відповідей на деякі з питань необхідно ставити додаткові питання.

2. Побудова моделі.

Задача чітко поставлена, тепер потрібно побудувати для неї математичну модель. Це дуже важливий крок у процесі розв'язання задачі і його слід добре обдумати, оскільки вибір моделі суттєво впливатиме на подальші кроки розробки алгоритму.

Як ви можете здогадатись, неможливо запропонувати набір формальних правил, що автоматизують стадію моделювання. Більшість задач моделюються індивідуально. Але разом з тим існує ряд принципів, яких слід дотримуватись при моделюванні. Вибір моделі – у переважній більшості все ж мистецтво, ніж наука, тому вивчення вдалих моделей – найкращий спосіб набути досвіду у моделюванні.

Приступаючи до розробки моделі, слід поставити хоча б 2 наступні питання:

1. Які математичні структури найбільш підходять для задачі?
2. Чи існують розв'язані аналогічні задачі?

Друге питання, можливо, є найкориснішим у всій математиці. В контексті моделювання воно досить часто дає відповідь на перше питання. Дійсно, існують задачі, що є модифікаціями раніше уже розв'язаних задач. Перш за все слід розглянути перше питання. Ми повинні описати математично, що відомо і що потрібно знайти. На вибір відповідних структур будуть мати вплив наступні фактори:

1) обмеженість знань розробника відносно невеликою кількістю структур;

2) зручність представлення;

3) простота обчислень;

4) корисність операцій, пов’язаних із даною структурою.

Здійснивши пробний вибір математичної структури задачі, слід переформулювати в термінах відповідних математичних об’єктів. Це буде однією із можливих моделей, якщо вдасться дати відповіді на наступні питання:

Чи уся важлива інформація задачі добре описана математичними об’єктами?

Чи існує математична величина, яка пов’язана із результатом?

Чи було виявлено корисні співвідношення між об’єктами у моделі?

Чи можна працювати із моделлю? Чи зручно з нею працювати?

3. Розробка алгоритму.

Як тільки задача чітко сформульована та для неї побудована модель, слід розпочинати роботу над розробкою алгоритму її розв’язання. Вибір методу, що достатньо часто залежить від вибору моделі, може у значній мірі вплинути на ефективність роботи алгоритму. Два різних алгоритми можуть бути правильними, але дуже відрізнятись за ефективністю.

4. Правильність алгоритму.

Кажуть, що алгоритм є коректним, якщо для кожного входу результатом його роботи є коректний вивід. Тоді коректний алгоритм розв’язує дану обчислювальну задачу. Якщо алгоритм некоректний, то для деяких входів він може взагалі не завершити свою роботу або видати відповідь, яка відрізняється від очікуваної.

Доведення правильності алгоритмів – це одна із найскладніших задач математичної науки в цілому. На даний час не існує єдиного ефективного методу доведення правильності алгоритмів. Є засоби, які дозволяють довести коректність деяких класів нескладних алгоритмів.

Ймовірно найбільш поширеною методикою доведення коректності алгоритмів є прогон їх на різних наборах тестів. Якщо вихідні дані алгоритму можна підтвердити наборами вхідних та вихідних даних, які обчислені вручну або є відомими, то виникає бажання сказати, що алгоритм працює. Але все ж існує ймовірність того, що алгоритм не запрацює.

Варто також зауважити, що правильність алгоритму ще нічого не говорить про його ефективність.

5. Реалізація алгоритму.

Як тільки описано алгоритм, наприклад, у вигляді послідовності кроків, та є підтвердження його коректності, то слід переходити до етапу його реалізації. Цей крок може бути достатньо складним та громіздким. По-перше, складність полягає у тому, що досить часто окремо взятий крок алгоритму може бути виражений у формі, які важко перевести безпосередньо у конструкцію мови програмування. По-друге, реалізація виявитись складним процесом, тому що перед написанням програми слід побудувати цілу систему структур даних для представлення важливих аспектів моделі, що використовується у задачі.

Аби зробити це, слід дати відповіді на наступні питання:

Які основні змінні?

Яких вони типів?

Скільки потрібно, наприклад, масивів та якої вони розмірності?

Чи є потреба у використанні зв'язаних списків?

Які підпрограми можна використати?

Якою мовою програмування доцільно користуватись?

Конкретна реалізація буде суттєво впливати на вимоги до пам'яті та швидкості алгоритму.

Наступний аспект побудови програмної реалізації – це програмування зверху-вниз. Цей підхід до розробки та реалізації алгоритмів та програм буде розглядатись дещо пізніше.

Слід також зауважити, що важливо доводити як правильність алгоритмів, записаних у словесній формі, так і правильність програм, побудованих на їх основі. З цієї причини потрібно ретельно слідкувати за тим, аби процес перетворення алгоритму на програму «заслуговував на довіру».

6. Аналіз алгоритму та його складності.

Існує ряд важливих практичних причин для аналізу алгоритмів. Однією із них отримання оцінок чи границь для об'єму пам'яті чи часу роботи, які використовуються алгоритмом для обробки конкретних даних. Машинний час та машинна пам'ять – достатньо важливі ресурси, на які одночасно можуть претендувати багато користувачів. Тому якісний аналіз алгоритмів повинен виявити ті розділи програми чи алгоритму, на які затрачається більша кількість ресурсів.

Існують також важливі і теоретичні причини для аналізу алгоритмів. Доцільно було б мати критерій, який дозволить порівняти два алгоритми, які розв'язують одну і ту ж задачу.

Поняття задачі. Під масовою задачею (або просто задачею) розуміють деяке загальне питання, на яке слід дати відповідь. У загальному випадку задача містить декілька параметрів або вільних змінних, конкретні значення яких не є визначеними. Задача N визначається наступною інформацією:

1. Загальним списком усіх її параметрів.

2. Формулюванням тих властивостей, яким повинна задовольняти відповідь або, іншими словами, розв'язання задачі.

Індивідуальна задачі I отримується із загальної задачі N, якщо усім параметрам задачі N присвоїти конкретні значення.

Розглянемо класичну задачу про комівояжера. Параметри цієї масової задачі складаються із набору «міст» C {c1,c2... cn} та відстаней d(ci, cj) між кожною парою міст ci та c j із C. Розв'язанням цієї задачі є такий впорядкований набір cp1, cp2 ... cpn заданих міст, який мінімізує величину

$$\sum_{i=1}^{n-1} d\left(c_{pi}, c_{pi+1}\right) + d\left(c_{pn}, c_{p1}\right)$$

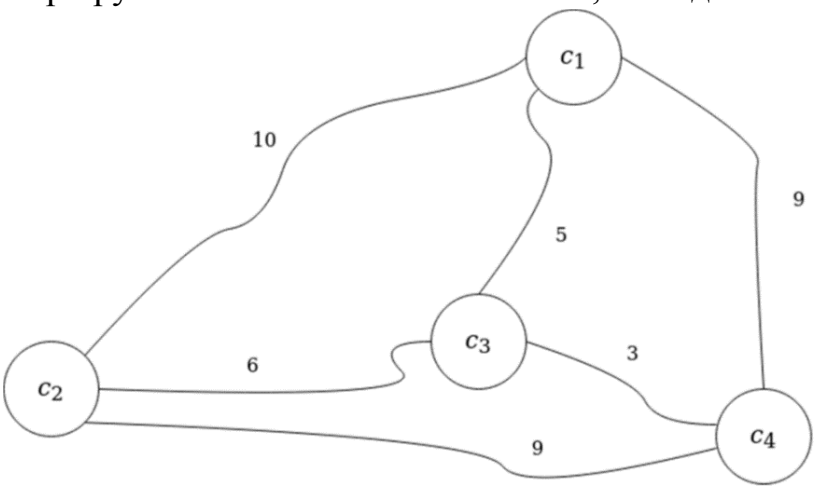
Цей вираз дає описує маршруту, який розпочинається у місті ср1, проходить послідовно через усі міста та повертається у вихідний пункт із міста cpn.

Індивідуальна задача комівояжера задається наступним чином (Рис. 1):

*C {c*1,*c*2... *c4}, d(c*1, *c*2*)* = 10, *d(c*1, *c*3*)* = 5, *d(c*1, *c*4*)* = 9, *d(c*2, *c*3*)* = 6, *d(c*2, *c*4*)* = 9,

*d(c*3, *c*4*)* = 3.

Послідовність {*c*1, *c*2, *c*4, *c*3} являє собою розв'язання задачі, оскільки такий маршрут є мінімальним із можливих, його довжина становить 27.



Блок-схема алгоритму задачі комівояжера представлена у презентації.

Контрольні питання

1. Дайте означення задачі.
2. У чому полягає різниця між масовою та індивідуальною задачами?
3. Наведіть приклад індивідуальної та масової задач.
4. Що таке алгоритм?
5. Назвіть основні властивості алгоритму.
6. Назвіть основні етапи розробки алгоритму.
7. Як здійснюється перевірка правильності роботи алгоритму?
8. Використання яких ресурсів оцінюється при аналізі алгоритму?
9. Які методи використовуються при розробці моделі розв'язуваної задачі?
10. Що є важливішим при розв'язуванні задачі: алгоритм чи структура даних?

Лекція 2. Способи представлення алгоритмів

2.1 Словесний та графічний способи

Для опису алгоритмів людина часто користується природною мовою, але для запису багатьох алгоритмів вона виявилась не зовсім зручною, тому виникла необхідність у створенні штучних мов, наприклад мови математичних формул для того, аби описати послідовність дій у алгоритмі. Найбільшого поширення для запису логічної структури алгоритмів отримали словесний спосіб, графічний (структурні схеми), псевдокод та мова програмування.

Словесна форма запису – це опис алгоритмів на природних мовах. Цей спосіб не так широко розповсюджений через його непрактичність, громіздкість.

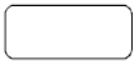
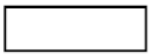
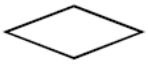
Розглянемо опис алгоритму пошуку максимального числа серед трьох заданих (числа не рівні між собою) {a, b, c}.

Крок 1. Якщо число a більше від числа b, то перейти до Крок 2. В іншому випадку перейти до Крок 3.

Крок 2. Якщо число a більше від числа c, то максимальне число a. В іншому випадку максимальне число c.

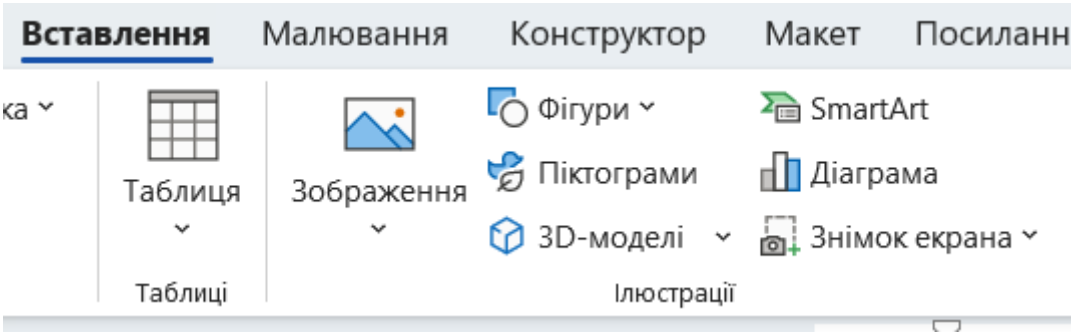
Крок 3. Якщо число b більше від числа c, то максимальне число b. В іншому випадку максимальне число c.

Графічна схема (блок-схема) алгоритму – це графічне зображення алгоритму у вигляді спеціальних блоків з необхідними словесними поясненнями. Кожний етап алгоритму представляється у вигляді геометричної фігури (блоку), що має певну форму в залежності від характеру операції. Блоки на схемі з’єднуються стрілками (лініями зв’язку), які визначають послідовність виконання операцій та утворюють логічну структуру алгоритму. Основні блоки графічної схеми продемонстровані нижче:

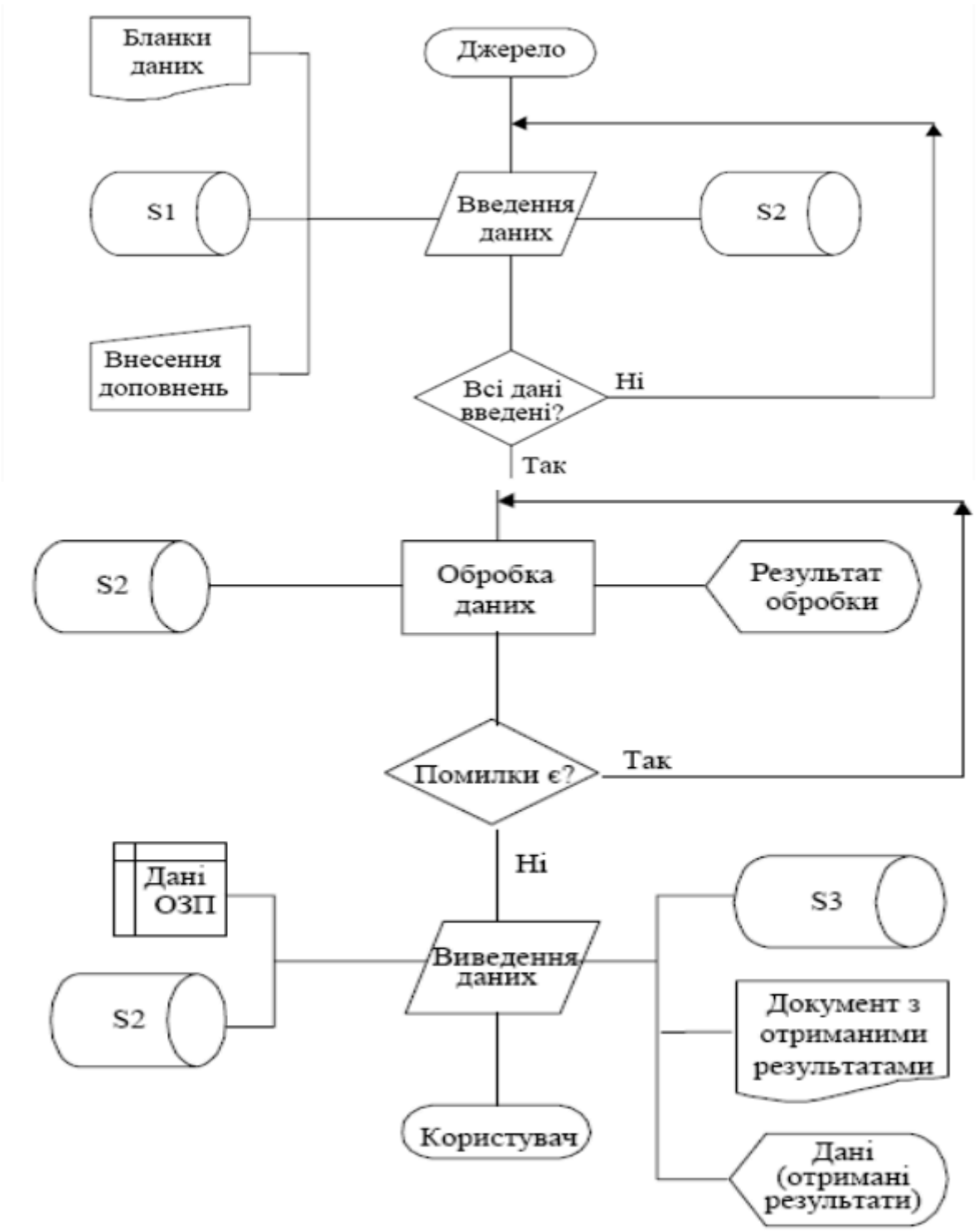
Позначення	Найменування	Опис
	блок пуск-зупинка	визначає початок та кінець алгоритму (для блоку пуск (початок) - визначений тільки один вихід, для блоку зупинка (кінець) – тільки вхід);
	блок введення-виведення	визначає введення інформації в програму або виведення на пристрій
	блок процес	визначає зміну значення, форми уявлення або розташування даних
	блок перевірки умови	визначає подальші кроки виконання алгоритму в залежності від виконання умови

У практично всіх програмах MS- Office є можливість використовувати блоки графічного уявлення.

У верхньому меню натискаємо Вставлення – Фігури – Блок-схема.



Тут пропонуються додаткові символи, які використовуються для позначень процесів інформатизації або при схематичному зображенні модульного програмування.



Важливою особливістю зображень базових структур алгоритмів є те, що вони мають один вхід і один вихід, що дозволяє при відносній незалежності конструювати окремі блоки алгоритмів, а потім окремо розроблені структури з’єднувати між собою (вихід однієї базової структури сполучається із входом іншої). Весь алгоритм представляє лінійну послідовність базових структур.

На будь-якій стадії існування алгоритми та програми представляють за допомогою конкретних графічних засобів, склад і правила вживання яких утворюють конкретні способи або форми запису.

Алгоритм пошуку максимального числа серед трьох заданих, представлений у вигляді блок-схеми, буде мати наступний вигляд:

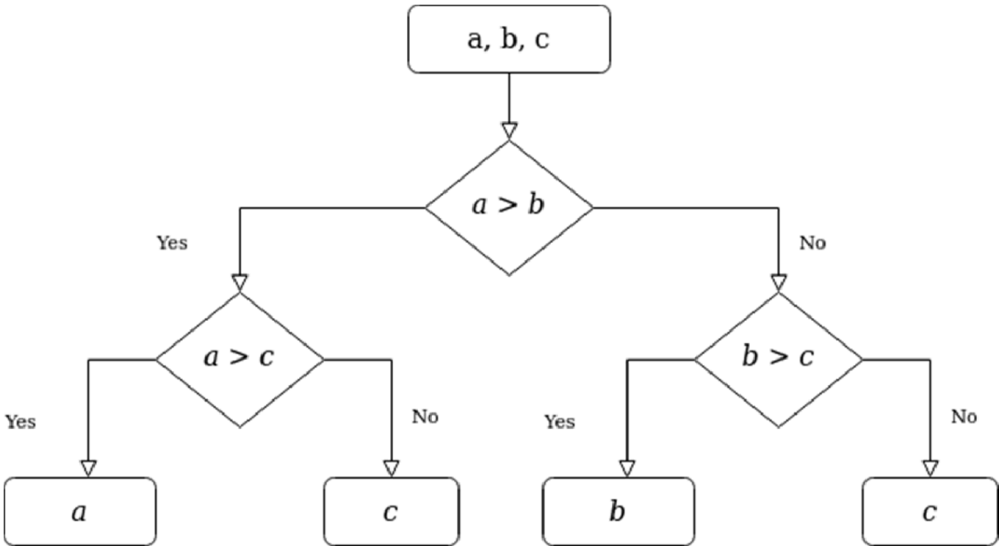


Рисунок 3 – Блок-схема алгоритму пошуку максимального числа серед трьох заданих

2.2 Поняття псевдокоду

Поряд зі схемами для зображення алгоритмів широко використовується псевдокод. **Псевдокодом** називається система правил запису алгоритму з використанням набору певних конструкцій для опису керуючих дій.

Псевдокод дозволяє формально зображати логіку алгоритму, використовуючи стандартизовані конструкції природної мови для зображення управління та зберігаючи можливості мови для опису дій по обробці інформації.

Даний спосіб тісно пов’язаний зі структурним підходом до програмування. Псевдокод займає проміжне положення між природною мовою та мовою програмування. Його застосовують переважно для того, щоб детальніше пояснити роботу програми, що полегшує перевірку правильності програми. Крім того, псевдокод дає розробнику велику свободу в зображенні алгоритму. Потрібно тільки використовувати стандартні управляючі конструкції та правила запису.

Чому потрібно використовувати псевдокод?

У великих проектах без псевдокоду можна просто загубитися. Написання псевдокоду дозволяє заздалегідь продумати потенційні проблеми. Ви отримуєте можливість спостерігати чисту логіку та порядок виконання програми, не переймаючись можливими помилками в синтаксисі. Створення псевдокоду перед написанням цього коду допомагає швидше завершувати проекти. Ви можете вважати його за ескіз своєї програми. Маючи перед очима псевдокод, легше зрозуміти, де і що має бути і як воно все має

працювати разом. Тому, коли ви приступите до етапу справжнього збирання, вам не доведеться довго роздумувати над тим, що необхідно робити, адже ви заздалегідь це визначили.

Головна перевага в тому, що псевдокод не залежить від будь-якої мови програмування. Логіка, написана вами, може бути застосована будь-ким і реалізована будь-якою мовою. Це дає свободу повторного використання та покращення архітектури створюваного вами додатка. Якщо ви не програміст, завдяки псевдокоду ви можете поділитися принципом та логікою роботи своєї майбутньої програми з кодерами. Тож вам буде простіше пояснити, що саме ви від них вимагаєте. Псевдокод можна писати його у будь-якому форматі, у тому числі з використанням академічного формату. Він добре структурований і деталізований, але пов'язані з застосуванням математики у великих кількостях. Або ви можете написати просто коротку схему того, що очікуєте від свого коду.

Алгоритм пошуку максимального числа серед трьох заданих, представлений у вигляді псевдокоду, буде мати наступний вигляд.

```
begin
a, b, c, max
if a > b then
if a > c then max := a else max := c
else
if b > c then max := b else max := c
end
```

Подивіться на приклад простої програми. Уявіть, що програма повинна замінити комбінацію букв "foo" в текстовому файлі. Програма прочитає кожен рядок у цьому файлі, шукатиме потрібну комбінацію в кожному рядку та замінюватиме її на іншу. Повторювані кроки починаються з пробілів - в ідеалі так має бути в реальному коді. Початковий ескіз псевдокоду може виглядати так:

- Відкрити файл
- У кожному рядку файлу:
    - Знайди комбінацію
    - Видаліть комбінацію
    - Вставка різних комбінацій
- Закрити файл

Стандартний процес псевдокодування

Напишіть лише одне звернення в рядку. Кожен крок в псевдокоді повинен давати комп'ютеру тільки одну дію. Найчастіше, якщо завдання правильно описане, кожному завданню буде відповідати один рядок псевдокоду. Напишіть список завдань, потім перетворіть його в псевдокод, а потім перетворіть псевдокод в реальний виконуваний код.

Список завдань:

Прочитати ім'я, вартість за годину, кількість годин

Виконуємо розрахунки

Сума нарахування = вартість години * кількість годин

Відрахування = Сума нарахування * Коефіцієнт відрахування

Сума після відрахування = Сума нарахування - відрахування

Ім'я запису, сума нарахування, відрахування, сума після відрахування

Псевдокод:

READ Ім'я, Значення години, Кількість годин, Коефіцієнт відрахування

Сума відрахування = Вартість годин * Кількість годин

Вирахування = Сума вирахування * Коефіцієнт відрахування

Сума після відрахування = Сума до відрахування – відрахування

WRITE ім'я, суму до відрахування, суму відрахування, суму після відрахування

Важливими ключовими словами можуть бути: READ, WRITE, IF, ELSE, ENDIF, WHILE, ENDWHILE, REPEAT и UNTIL.

2.3 Мови програмування для опису алгоритмів

Останнім способом запису алгоритмів є **мова програмування**. Розглянуті вище способи зручні для програміста, але не прийнятні для комп’ютера, оскільки вони не можуть бути однозначно зрозумілі.

Мова програмування – це знакова система, призначена для опису процесів вирішення завдань та їх реалізації із використанням обчислювальної техніки.

Реалізація означає, що описи можуть бути задані комп’ютеру і однозначно ним зрозумілі. До мов програмування відносяться мови команд або машинні мови та мови високого рівня.

Перша група являє власну мову обчислювальної машини і виконання програми можливе лише в тому випадку, якщо вона записана на цій мові. Однак програмувати на машинному мові досить важко, що обумовлено надмірною деталізацією програми, необхідністю знати конкретну систему команд і детально представляти роботу. Представлення складної програми на машинній мові незручно для сприйняття людиною. Ці недоліки послужили стимулом для створення мов програмування високого рівня, які не збігаються з машинними. Ідея таких мов полягає в представленні програм у вигляді не тільки прийнятному для комп’ютера, а й зручному для користувача. Комфортність означає, що спосіб запису повинен відображати основні ідеї програмування і представляти програму в однозначною, природною, у формі, що легко сприймається. Це означає, що програма може бути введена задана комп’ютеру і однозначно зрозуміла, тобто однозначно переведена на машинну мову. Мови програмування високого рівня дають розробнику велику свободу в конструюванні програм, але не звільняють його від необхідності враховувати той факт, що саме комп’ютер буде виконувати його програму і що саме комп’ютер накладає на програму обмеження, обумовлені скінченністю її швидкості та пам’яті.

Алгоритм пошуку максимального числа серед трьох заданих, представлений у вигляді програми на мові програмування С++, буде мати наступний вигляд.

```
int a, b, c, maxNum;
if (a > b) {
if (a > c) {maxNum = a;}
else {maxNum = c;}
}
else {
if (b > c) {maxNum = b;}
else {maxNum = c;}
}
cout<<maxNum;
```

Контрольні питання

1. Назвіть основні способи представлення алгоритмів.
2. Назвіть переваги та недоліки словесного способу представлення.
3. Назвіть переваги та недоліки представлення алгоритмів у вигляді блок-схеми.
4. Назвіть переваги та недоліки представлення алгоритму у вигляді псевдокоду.
5. Назвіть переваги та недоліки представлення алгоритму у вигляді програмного коду.

Лекція 3. Структура програмного забезпечення ПК. Загальні відомості про операційні системи, основні функції ОС

3.1 Історія розвитку операційних систем

Перші комп'ютери взагалі не мали ОС. На початку 1960-х років вони лише комплектувались набором інструментів для розроблення, планування та виконання завдань. Серед інших можна виділити системи від UNIVAC та Control Data Corporation.

До кінця 1960-х років було розроблено цілий ряд операційних систем, в яких були реалізовані всі або більшість з вищеперелічених функцій. До них можна віднести «Atlas» (Манчестерський університет), «CTTS» і «ITSS» (Массачусетський технологічний інститут), «THE» (Ейндховенський технологічний університет), «RS4000» (Університет Орхуса) та інші (на той момент їх налічувалось близько сотні).

Найрозвинутіші ОС того часу «OS/360» (компанія «IBM»), «SCOPE» (компанія «CDC») та завершений вже в 1970-х роках «MULTICS» (MTI та

компанія «Bell Labs»), передбачали можливість використання багатопроцесорних системи.

Спонтанний характер розроблення ОС призвів до наростання кризових явищ, пов'язаних зі складністю та великими розмірами розроблюваних систем. ОС погано масштабувались (простіші не використовували всіх можливостей потужних ПК; складніші неоптимальне виконувались або взагалі не виконувались на менш потужних системах) і були повністю несумісними між собою.

У 1969 році співробітники MTI Кен Томпсон, Деніс Рітчі та Брайан Керніган з колегами розробили та реалізували ОС «Юнікс» («Unix»; первинно «UNICS», на противагу «MULTICS»), яка увібрала в себе багато рис попередниць, але на противагу їм мала цілий ряд переваг:

- проста метафорика (два ключових поняття — процес та файл);
- компонентна архітектура (принцип «одна програма — одна функція», або інакше «кожна програма має робити лише одну роботу, але робити її добре» плюс потужні засоби об'єднання цих програм для вирішення конкретних задач);
- мінімізація ядра та кількості системних викликів;
- незалежність від апаратної архітектури і реалізація машинно незалежною мовою програмування (для цього була розроблена мова програмування «C»;
- уніфікація файлів (будь-що у системі є файлом, до якого можна доступитись спільними для всіх правилами).

Завдяки зручності перш за все в якості інструментального середовища «Юнікс» дуже тепло зустріли в університетах, а потім і в галузі в цілому і незабаром вона стала прототипом єдиної ОС, яку можна було використовувати у найрізноманітніших інформаційних системах, і — більше того — швидко та з мінімумом зусиль перенести на іншу апаратну архітектуру.

Незабаром «Юнікс» стала стандартом де-факто, а потім і юридичним — ISO/IEC 9945. ОС, що дотримувались цього стандарту чи опираються на нього, називають «відкритими» або «стандартними». До них відносять системи, що базуються на останній версії «Юнікс», випущеної «Bell Labs» («System V»), на розробках Університету Берклі («FreeBSD», «OpenBSD», «NetBSD»), а також ОС «Linux», розроблена спільнотою на чолі з Лінусом Торвальдсом та в межах проекту «GNU» (основні системні інструменти).

Сучасні операційні системи типове мають графічний інтерфейс користувача, який на додачу до клавіатури користується також вказівниковим пристроєм — мишею чи тачпадом. Старіші системи, та системи, що не призначені для частої безпосередньої взаємодії з користувачем (як наприклад сервери) типове використовують інтерфейс командного рядка. Обидва підходи так чи інакше реалізують оболонку, яка перетворює команди користувача — текстові з клавіатури, чи рухи мишки — на системні виклики.

3.2 Поява та розвиток ОС Microsoft Windows

Спочатку родина ОС Microsoft Windows проектувалась як графічна надбудова над старими середовищами DOS. Сучасні версії розроблені на базі нового ядра (NT - New Technology, Нова технологія), яке з'явилось в OS/2, запозичене з VMS. Windows запускається на 32- та 64-бітних процесорах Інтел та AMD.

Створення сімейства продуктів Windows розпочалося 1983 року, проте лише 1995 року було випущено першу повноцінну операційну систему Windows 95. Це була також перша 32-розрядна ОС від Майкрософт, її нащадками стали ОС Windows 98 і Windows ME. Компанія Майкрософт розвивала й лінійку серверних операційних систем, які отримали назву Windows NT (Windows New Technology — нова технологія Windows). Перша версія, Windows NT 3.1, з'явилася 1993 року. До цієї лінійки належать також Windows 2000, Windows XP, Windows Vista, Windows Server 2003 та Windows Server 2008.

До початку 1990-х років найпопулярнішою ОС для персональних комп'ютерів була система MS-DOS від Майкрософт, яка мала текстовий інтерфейс. Звичайно, історія ОС не обмежується продуктами корпорації Майкрософт. Завжди існувала альтернатива комерційним системам Майкрософт, насамперед це сімейство серверних систем із відкритим кодом UNIX і Linux. Термін «із відкритим кодом» означає не лише те, що система поширюється вільно (безкоштовно), а й те, що її вихідний код доступний усім бажаючим. Якщо розглядати не лише IBM-сумісні персональні комп'ютери, то набір операційних систем стає значно ширшим: для комп'ютерів фірми Apple існує власна ОС — Mac OS, для мейнфреймів IBM — система z/OS, для мобільних пристроїв — Windows Mobile і Symbian.

Станом на 2006 рік Windows утримує монопольне становище (близько 94%) світового ринку настільних систем, дещо втрачаючи позиції через зростання популярності систем з відкритими джерельними кодами. Вона також використовується на малих та середніх серверах мереж та баз даних. Останнім часом Microsoft проводить ряд маркетингових досліджень, які мають на меті показати привабливість родини Windows на ринку корпоративних систем.

У листопаді 2006 року, після більш ніж 5 років розробки, корпорація Microsoft випустила ОС Windows Vista, що містить велику кількість нововведень та архітектурних змін в порівнянні з попередніми версіями Windows. Серед інших можна виділити новий інтерфейс користувача, названий Windows Aero, ряд вдосконалень безпеки, як наприклад Контроль реєстраційного запису користувача (User Account Control) та нові програми для мультимедія, як наприклад Windows DVD Maker.

3.3 Особливості операційної системи Windows

ОС Windows – це програма, яка забезпечує зручне середовище, в якому працює користувач.

Операційна система Windows має наступні особливості:

- Інтерфейс користувача:
- Наявність кнопки «Пуск», яка забезпечує швидкий доступ до програм та документів, з якими останнім часом працював користувач;
- Відображення панелі задач на екрані в будь-який момент;
- Наявність спеціального значка «Мережеве оточення», який забезпечує можливість в будь-який час переглянути стан мережі;
- Можливість створювати ярлики для файлів, документів, дисків, комп'ютерів;
- Можливість використовувати для будь-якого об'єкта контекстне меню, що містить розділ «Властивості»;
- Наявність набору засобів для швидкого переглядання документів;
- Наявність потужних засобів для пошуку;
- Наявність «Кошику»;
- Наявність допоміжних інструментів для роботи з принтерами та шрифтами;
- Наявність колективу «Майстрів».
- Підтримання довгих імен файлів (до 255 символів).
- Підтримання технології plug-and-play. Ця технологія передбачає можливість ОС самостійно, без додаткового перезавантаження комп'ютера визначити всі апаратні складові, що підключені до системи. При цьому вона самостійно шукає в своїй базі даних (що знаходиться в BIOSi), необхідні для кожної апаратної частини драйвери.
- *32 – розрядна операційна система.*

Забезпечує можливість працювати в попередній версії ОС MS-DOS (чи іншої ОС, якщо Windows інсталювався на її основі). MS-DOS 7.0 – частина Windows. Забезпечує можливість працювати з іншими операційними системами (OS/2, UNIX, Windows NT).

Багатозадачна ОС (при одночасному розв'язанні кількох задач ОС відводить для кожної програми окрему, «віртуальну машину» (область пам'ять, яку програма сприймає як повні ресурси комп'ютера). Для 16- і 32-розрядних додатків Windows надає відповідно одну (спільну для всіх машину).

- Підтримує режим багатопоточності.
- Підтримує роботу комп'ютерної мережі.
- Перетягування drag and drop (перетягнути та кинути).

Windows можна встановити при наявності на комп'ютері однієї з ОС: OS/2, UNIX, Windows NT, MS-DOS.

Windows 10 проти Windows 11: порівняння продуктивності

Windows 11 — найновіша версія лінійки операційних систем Windows NT компанії Microsoft, випущена 24 червня 2021 року. Це безплатне оновлення своєї попередниці, Windows 10 (2015), доступне для будь-яких

пристроїв з Windows 10, що відповідають новим системним вимогам Windows 11.

У порівнянні з Windows 10, Windows 11 дійсно може потенційно мати кращу швидкість роботи. У Microsoft розповіли про переваги продуктивності і оптимізації у Windows 11 у відео, яке було опубліковано на їхньому офіційному каналі в YouTube. Загалом, переваги продуктивності у Windows 11 у значній мірі зводяться до того, як нова операційна система обробляє системні процеси, які можна з легкістю побачити просто відкривши диспетчер задач (можна відкрити за допомогою хоткею «Ctrl + Shift + Esc»).

Які помилки є у Windows 11

Windows 11 – це найновіша операційна система Microsoft, а Windows 10 існує вже п’ять років. Враховуючи це, можна очікувати, що у Windows 11 буде багато багів і проблем, які можуть вплинути на продуктивність вашої системи. У той же час, варто мати на увазі, що Microsoft кожного місяця будуть випускати оновлення для Windows 11, а тому весь цей масив проблем буде поступово вирішуватися. Хоча, з досвіду попередніх років, з кожним оновленням нас очікують і нові проблеми та баги.

Контрольні питання

1. Визначте, які основні етапи пройшли операційні системи від свого зародження до кінця 1960-х років.
2. Поясніть, які переваги мала операційна система Unix порівняно з попередніми ОС.
3. Охарактеризуйте, які ключові особливості Unix сприяли її поширенню та стандартизації.
4. Розкрийте, як розвивалась операційна система Windows та які її версії були найбільш знаковими.
5. Проаналізуйте, які відмінності існують між Windows NT та ранніми версіями Windows.
6. Визначте, які основні особливості графічного інтерфейсу Windows сприяли її популярності серед користувачів.
7. Порівняйте, які переваги та недоліки має Windows у порівнянні з UNIX-подібними системами.
8. Опишіть, які функції операційної системи Windows дозволяють користувачам ефективно працювати в багатозадачному режимі.
9. Поясніть, у чому полягає різниця між Windows 10 та Windows 11 з точки зору продуктивності та системних вимог.
10. Проаналізуйте, які основні помилки та проблеми виникають у Windows 11 та як вони можуть впливати на роботу користувача.

Лекція 4. Файлова система ПК. Диски, файли, папки. Імена файлів, папок. Шлях до файлу. Шаблони імен.

4.1 Файлова система персонального комп’ютера. Види файлових систем.

Файлова система (скорочено ФС) — це сукупність методів і структур, які використовуються операційною системою (ОС) комп'ютера для впорядкування даних на будь-якому цифровому запам'ятовувальному пристрої, а також для контролю над вільним простором для зберігання даних. Щоб краще зрозуміти сутність такої складної технології, слід ознайомитися з головними принципами в її основі.

Почнемо з того, що будь-який комп'ютерний файл зберігається на носії даних (наприклад, жорсткому диску, SSD, флеш-накопичувачі тощо), який має певну ємність. Цей носій можна розглядати як лінійний простір, доступний для читання або як для читання, так і для запису цифрової інформації. Кожен байт даних на ньому має своє зміщення (offset) від початку сховища, відоме як *адреса*, яка використовується для посилання на нього. У цьому відношенні сховище можна розглядати як сітку з набором *пронумерованих комірок* (кожна комірка — це один байт). І будь-який об'єкт, що зберігається до сховища, отримує власні комірки.

Комп'ютерні сховища зазвичай використовують пару *сектор* та *зміщення у секторі* для посилання на будь-який байт інформації, який вони містять. Сектор — це група байтів (зазвичай *512 байт* чи *4096 байт* у випадку нових накопичувачів великої ємності), яка є мінімальною адресною одиницею фізичного сховища. *Наприклад*, байт *1040* на жорсткому диску буде вказаний як сектор *#3* та *зміщення у секторі 16 байтів* ([сектор]+[сектор]+[16 байтів]). Ця схема застосовується для оптимізації адресації та використання меншого числа для звернення до будь-якої частини інформації, розміщеної у сховищі.

Щоб опустити другу частину адреси (зміщення у секторі), файли зазвичай зберігаються, *починаючи з початку сектора* та *займають цілі сектори* (наприклад: 10-байтовий файл займає весь сектор, 512-байтовий файл також займає весь сектор, тоді як 514-байтовий файл займає два повних сектори).

Кожен файл зберігається у *"невикористаних" секторах* і потім його можна прочитати, якщо відома його позиція та розмір. Але як ОС дізнається, які сектори зайняті, а які вільні? Де зберігаються розмір, позиція та ім'я файлу? Саме за це відповідає файлова система.

В цілому **файлова система** є структурованим представленням даних з набором **метаданих**, які описують ці дані. Вона застосовується до сховища під час його форматування. Ця структура служить для всього сховища, а також є частиною його ізольованого сегмента – **розділу диска**. Зазвичай вона оперує *блоками*, а не секторами. **Блоки ФС** — це групи секторів, за допомогою яких оптимізується адресація сховища.

Крім виконання вищезазначених функцій, ФС також відповідає за:

- Виділення блоків для нових файлів;
- Присвоєння імен та інших важливих властивостей файлам;
- Групування файлів у каталоги;
- Читання існуючих файлів та внесення записів у них;
- Виконання видалення файлів.

Постійні *операції запису/видалення* даних у сховищі можуть спричинити його **фрагментацію**. Тобто файли не зберігаються цілими одиницями, а розбиваються на фрагменти. При роботі з файлом користувач надає великі обсяги текста чи фото. Файл значно збільшується. Та вже не може займати те місто на диску, в якому він був розміщене. Тоді частина файлу, або колекції фото зберігається в іншому місті. Інформація про обидва фрагменти як частини одного файлу зберігається у файловій системі.

Проте не всі файлові системи однакові. Вони можуть істотно відрізнятися за своїми стратегіями організації даних, а також за такими характеристиками, як продуктивність, стабільність та надійність. Деякі з них також можуть підходити лише для окремих випадків застосування.

4.2 Диски, файли, папки.

Диск – це форматована для певної файлової системи область зберігання даних, якої призначена буква.

Для зберігання може використовуватися гнучкий диск, компакт-диск, жорсткий диск або частина ЖД, SSD – диск, флеш-пам'ять, карти пам'яті або пристрої іншого типу.

Файл – поіменована область на диску, що вміщує інформацію, текст, графіку, відео і т.ін.; це іменована сукупність будь-яких даних, що зберігається, пересилається та обробляється як єдине ціле.

Файл – одиниця збереження інформації, яка обов'язково має ім'я і може мати будь-який розмір інформації (максимальний розмір залежить від файлової системи). В деяких операційних системах (Unix) файл може мати більш ніж одне ім’я (жорсткі посилання).

Розширення файлу - послідовність символів, що додаються до імені файлу і призначені для ідентифікації типу файлу. Це один з найпоширеніших методів, за допомогою яких користувач або програмне забезпечення комп'ютера може визначити тип даних, що зберігаються у файлі.

В деяких файлових системах, таких як NTFS, передбачені атрибути. Практично атрибути не впливають на можливість доступу до файлів, для цього в деяких файлових системах існують права доступу.

В різних операційних та файлових системах можуть бути реалізовані різні типи файлів; крім того, реалізація різних типів може відрізнятися, як це представлене в таблиці 4.1.

Таблиця 4.1 – Типи та характеристики файлів

Тип файлу	Характеристика

Звичайний файл	файл, що дозволяє операції читання, запису, переміщення всередині файлу
Каталог або директорія / папка	файл, що містить записи про файли, що до нього входять. Каталоги можуть містити записи про інші каталоги, утворюючи деревоподібну структуру
Жорстке посилання	в загальному випадку, одна і та ж область інформації може мати кілька імен. Після створення жорсткого посилання сказати де «справжній» файл, а де жорстке посилання неможливо, тому що імена рівноправні. Жорсткі посилання можливі тільки на одному фізичному носії
Символьне посилання	файл, що містить в собі посилання на інший файл або директорію. Може посилатися на будь-який елемент файлової системи, у тому числі, і розташований на іншому фізичному носії

Практично завжди файли на дисках об'єднуються в каталоги.

Папка(каталог) – спеціальне місце на диску, що має своє ім’я, вміщує в собі файли й папки і відомості про них.

Папка – це спеціальне місце на диску для зберігання відомостей про файли та інші папки

Файли об'єднуються в папках за будь-якою ознакою, заданою користувачем.

Каталоги на різних дисках можуть утворювати кілька окремих дерев, як в DOS/Windows, або ж об'єднуватися в одне дерево, спільне для всіх дисків, як в UNIX-подібних системах.

У UNIX існує тільки один кореневий каталог, а всі інші файли та каталоги вкладені в нього. У більшості UNIX-подібних систем знімні диски (дискети і CD), флеш-накопичувачі і інші зовнішні пристрої зберігання даних монтують в каталог /mnt, /mount або /media.

Слід звернути увагу на використання слешів у файлових системах Windows, UNIX і UNIX-подібних операційних системах (У Windows використовується зворотний слеш «\», а в UNIX і UNIX-подібних операційних системах простий слеш «/»).

Один файл може містити кілька варіантів вмісту. Таким чином, в одному файлі можна зберігати декілька версій одного документу, а також додаткові дані (значок файлу, пов'язаного з файлом програми). Така організація типова для HFS на Macintosh.

Класифікація файлових систем

В залежності від організації файлів на носії даних, файлові системи можуть поділятись на:

- ієрархічні файлові системи — дозволяють розміщувати файли в каталоги;
- плоскі файлові системи — не використовують каталогів;

- кластерні файлові системи — дозволяють розподіляти файли між кількома однотипними фізичними пристроями однієї машини;
- мережеві файлові системи — забезпечують механізми доступу до файлів однієї машини з інших машин мережі;
- розподілені файлові системи — забезпечують зберігання файлів шляхом їх розподілу між кількома машинами мережі.

За призначенням файлові системи можна класифікувати на такі категорії:

- Для носіїв з довільним доступом (наприклад, жорсткий диск): FAT32, HPFS, ext2. Останнім часом поширилися журнальовані файлові системи, такі як ext3, ext4, ReiserFS, JFS, NTFS, XFS.
- Для носіїв з послідовним доступом (наприклад, магнітні стрічки): QIC.
- Для оптичних носіїв — CD і DVD: ISO 9660, HFS, UDF.
- Віртуальні файлові системи: AEFS і ін.
- Мережеві файлові системи: NFS, SMBFS, SSHFS, Gmailfs.

FAT (File Allocation Table) - таблиця розміщення файлів. Існує декілька версій FAT, а саме FAT12, FAT16, FAT32, exFAT (FAT64). Число в абревіатурі вказує розмір елемента таблиці в бітах. Одиницею розподіленої пам'яті є кластер.

Файлова система FAT32 була вперше реалізована в операційній системі Windows 95. FAT32 підтримує томи (логічні диски) обсягом до 8 ТБ.

Не можна створити в розділі FAT32 файл, розмір якого перевищує значення (2^32)-1 байт (на один байт менше, ніж 4 ГБ).

exFAT (Extended FAT) — файлова система, призначена для флеш-накопичувачів.

Основними перевагами exFAT перед попередніми версіями FAT є:

- Зменшення кількості перезаписів одного і того ж сектора. Це було основною причиною розробки ExFAT.
- Теоретичний ліміт на розмір файлу 264 байт (16 ексбібайт).
- Поліпшення розподілу вільного місця за рахунок введення біт-карти вільного місця, що може зменшувати фрагментацію диска.

NTFS (New Technology File System) — стандартна файлова система для сімейства операційних систем Microsoft Windows NT.

NTFS замінила файлову систему FAT, яка використовувалася в MS-DOS і попередніх до Windows NT версіях Microsoft Windows. NTFS підтримує систему метаданих і використовує спеціалізовані структури даних для зберігання інформації про файли для поліпшення продуктивності, надійності і ефективності використання дискового простору. NTFS використовує систему журналювання для підвищення надійності файлової системи. У Файловій системі NTFS відсутнє розділення на атрибути.

Журнальована файлова система — файлова система, що зберігає інформацію про всі зроблені зміни на диску в спеціальному журналі. Зміни завжди зберігаються перед їх застосуванням до основної файлової системи і у разі виникнення збою системи або відключення живлення, відновлення таких

файлових систем проходить швидше і існує менша ймовірність пошкодження.

ext3 — журнальована файлова система, що використовується в операційних системах на ядрі Linux, є файловою системою за замовчанням у багатьох дистрибутивах. Файлова система ext3 може підтримувати файли розміром до 1 ТБ.

HFS+ - файлова система, розроблена Apple Inc. для заміни раніше використовуваної HFS, основної файлової системи на комп'ютерах Macintosh.

APFS (Apple File System) - файлова система, розроблена Apple Inc. для заміни попередньої файлової системи HFS+. Apple File System – це нова, сучасна файлова система, розроблена компанією Apple для використання в iOS, macOS, tvOS і watchOS. Ця файлова система, оптимізована для роботи з Flash/SSD накопичувачами.

Імена папок створюються за тими самими правилами, що й імена файлів, але без розширення крім / **\? : * | "<>**

Для вказівки точного розташування файлу вказується маршрут (шлях) до файлу, що проходить через систему вкладених папок з верхнього рівня

Як роздільник використовується \

Диск: \ вкладені папки \ ім'я файлу

D:\Кафедра економічної кібернетики та інформаційних технології\ЛЕКЦІЇ\Для стаціонару\2024\Презентація Лекції №1_ЕКІТ_2024. Pptx

4.3 Імена файлів, папок. Шлях до файлу. Шаблони імен.

Ім'я файлу складається з власного імені та розширення, що розділяються точкою.

Розширення вказує на тип (призначення) файлу і вказується користувачем або програмним середовищем, яке створило файл.

Розширення складається з **3 - 4 - х** букв

В ОС MS DOS було прийнято угоду 8.3 алфавітно-цифрового символу латинського алфавіту для імен файлів.

У Windows використовується "довге" ім'я файлу – 255 будь-яких символів (кирилиця, латинські літери, цифри, розділові знаки, пробіли) крім / **\? : * | "<>**

У довжину імені файлу включається шлях до файлу.

D:\Кафедра економічної кібернетики та інформаційних технології\ЛЕКЦІЇ\Для стаціонару\2024\Презентація Лекції №1_ЕКІТ_2024. pptx

Для пошуку групи файлів, об'єднаних загальними ознаками, вказується шаблон імені файла.

У шаблоні використовуються спеціальні символи:

*** та ?**

* - будь-яке число будь-яких символів у імені або розширенні файлів;

? - один довільний символ чи відсутність символу.

Приклад:

** .doc*

А.doc*

Лаб.doc

Документ – це об’єкт, створений користувачем, що вміщує дані: текст, таблиці, малюнок.

Ярлик-це посилання на об’єкт, що знаходяться у файловій системі, його створюють для швидкого доступу до об’єкта, його розмір 1кб.

Значок-це графічне представлення об’єкта, яке забезпечує визначення типу об’єкта.

Об’єкт-будь-який елемент ОС (програма, диск, документ), файл, ярлик, значок.

Контрольні питання

1. Визначте, що таке файлова система та які її основні функції.
2. Поясніть, як операційна система визначає зайняті та вільні сектори на цифровому носії.
3. Охарактеризуйте, що таке фрагментація файлів та як вона впливає на продуктивність файлової системи.
4. Розкрийте, які існують типи файлів та їхні характеристики в операційних системах.
5. Проаналізуйте, які основні види файлових систем використовуються сьогодні та які їхні особливості.
6. Визначте, у чому різниця між FAT32, exFAT та NTFS, а також для яких пристроїв вони найкраще підходять.
7. Порівняйте, які переваги має журнальована файлова система (наприклад, NTFS або ext3) порівняно з файловими системами без журналювання.
8. Опишіть, як організована структура каталогів у файлових системах Windows та UNIX.
9. Поясніть, що таке шлях до файлу та які особливості його запису в різних операційних системах.
10. Проаналізуйте, як працюють шаблони імен файлів та які символи використовуються для пошуку файлів за заданими критеріями.

Лекція 5. Поняття про базові алгоритмічні конструкції

5.1 Лінійна алгоритмічна конструкція.

Базові алгоритмічні конструкції або базові алгоритмічні структури – це основні структурні елементи, за допомогою яких створюють алгоритм для розв'язування деякої загальної задачі. Існують три основні (базові) алгоритмічні структури: *слідування* (або лінійна структура), *розгалуження* та *повторення* (або цикл), компонуючи які можна представити логічну структуру будь-якого алгоритму.

Основною особливістю базових алгоритмічних структур є їх повнота. Це означає, що цих структур достатньо для створення алгоритмів довільної складності, та наявність в них одного входу та одного виходу.

Структура слідування або лінійна алгоритмічна структура – це алгоритмічна структура, яка забезпечує отримання результату шляхом одноразового виконання дії, незалежно від вхідних даних та проміжних результатів. Дії у такій структурі виконуються послідовно, одна за одною, або лінійно.

На алгоритмічній мові структура слідування виглядає наступним чином:

Алгоритм:
Вхід
Дія 1
Дія 2
…
Дія N
Вихід

Алгоритми, які використовують лише структуру слідування, називаються **лінійними**. Варто зауважити, що усі алгоритми використовують структуру слідування, але не всі вони є лінійними.

5.2 Алгоритмічна структура розгалуження.

Структура розгалуження (структура вибору або умова) – вибір дії або групи дій (блоку) у разі виконання або невиконання заданої умови. Ця алгоритмічна структура залежно від результату перевірки умови (так чи ні, істина або хибно) забезпечу виконання одного з альтернативних шляхів роботи алгоритму. Кожен із них веде до загального виходу, тому робота алгоритму триватиме незалежно від того, який із варіантів було обрано.

Розгалуження поділяються на два типи: *повні і неповні*.

Неповне розгалуження – це розгалуження, в якому хід алгоритму визначено лише в разі виконання умови. Якщо умова не виконується, то робота алгоритму визначається алгоритмічною конструкцією, яка розміщена після розгалуження.

На алгоритмічній мові структура неповного розгалуження виглядає наступним чином:

якщо … то …

Алгоритм:

Вхід

…

якщо умова то дія

…

Вихід

Повне розгалуження – це розгалуження, в якому певні дії визначено у двох випадках: виконання та невиконання умови.

На алгоритмічній мові структура повного розгалуження виглядає наступним чином:

якщо … то … інакше

Алгоритм:

Вхід

…

якщо умова то дія1 інакше дія2

…

Вихід

Поширеним випадком є розміщення однієї структури вибору в іншій. Такі структури називають *вкладеними розгалуженнями*.

Структура розгалуження існує в чотирьох основних варіантах:

- якщо то;

- якщо то-інакше;

- вибір;

- вибір-інакше.

Вибір – це вказівка багатовибірного розгалуження, яка дає змогу вибрати одну з множини альтернатив. Така структура використовується у тому випадку, коли кількість вкладень у розгалуженнях є досить значною, що у свою чергу погіршує читабельність коду.

На алгоритмічній мові структура повного розгалуження виглядає наступним чином:

вибір

при умова 1: дії 1

при умова 2: дії 2

. . . . . .

при умова N: дії N

У тому випадку, якщо вхідні дані не задовольняють жодну із умов, яка запропонована у структурі вибору, керування ходом алгоритму передається до команди, що розміщена після вибору.

Вибір-інакше – це структура вибору, для якої запропоновано команду для випадку, коли вхідні дані не задовольняють жодну із запропонованих альтернатив.

На алгоритмічній мові структура вибору виглядає наступним чином:

вибір
при умова 1: дії 1
при умова 2: дії 2
. . . . . .
при умова N-1: дії N-1
інакше дії N

5.3 Цикли

Структура повторення (структура циклу або цикл) - це алгоритмічна структура, в якій передбачено повторення деякої вказівки або блоку вказівок. За допомогою цієї структури описують однотипні дії, що повторюються багатократно. Такі алгоритми забезпечують виконання значної послідовності дій, записаних порівняно короткою послідовністю команд.

Повторення забезпечує багаторазове виконання деякої сукупності команд, яку називають *тілом циклу*.

Існують три види циклів.

Цикл із лічильником (цикл із параметром) – це алгоритмічна конструкція, в якій кількість повторень задано заздалегідь, керується лічильником (ітератором) і логічними виразами, розташованими на його початку. Це єдиний вид циклу, для якого кількість повторень є наперед заданою.

На алгоритмічній мові структура циклу із параметром виглядає наступним чином:

початок циклу для і від і1 до і2
тіло циклу
кінець циклу

Цикл із передумовою – це алгоритмічна конструкція, що забезпечує виконання команди або групи команд доти, поки виконується (є істинною) логічна умова, сформульована на його початку. Для циклу із передумовою кількість повторень не є наперед заданою. Якщо вхідні дані забезпечують хибність логічної умови, яка задана на початку циклу, то його тіло може не виконатись жодного разу.

У такому випадку керування ходом алгоритму переходить до команди (вказівки), яка розміщена після циклу.

На алгоритмічній мові структура циклу з передумовою виглядає наступним чином:

початок циклу поки умова
тіло циклу
кінець циклу

Цикл із післяумовою – це алгоритмічна конструкція, що забезпечує виконання команди або групи команд доти, поки виконується (є істинною) логічна умова, сформульована у його кінці. Для циклу із післяумовою кількість повторень не є наперед заданою. Якщо вхідні дані забезпечують хибність логічної умови, яка задана в кінці циклу, то його тіло виконається

один раз. Після цього керування ходом алгоритму переходить до команди (вказівки), яка розміщена після циклу.

Тобто якщо тіло циклу з передумовою може не виконатись жодного разу, то тіло циклу з післяумовою обов'язково виконається хоча б один раз.

На алгоритмічній мові структура циклу з післяумовою виглядає наступним чином:

початок циклу
тіло циклу
поки умова кінець циклу

Теоретичним обґрунтуванням того, що для побудови алгоритму довільної складності достатнім є використання лише трьох алгоритмічних структур (слідування, розгалуження та циклу), є теорема Бьома-Якопіні.

Теорема Бьома-Якопіні: виконуваний алгоритм може бути втілено з використанням лише трьох конкретних керівних структур: послідовного виконання, розгалужень, повторень (циклів).

Цю теорему було сформульовано та доведено італійськими математиками Коррадо Бьомом і Джузеппе Якопіні в їхній статті у 1966 р. У статті було описано методи перетворення неструктурованих алгоритмів на структуровані на прикладі створеної Бьомом мови програмування P''.

Структурна теорема Бьома-Якопіні була початком структурного програмування – парадигми програмування, яка виключає команди безумовного переходу (goto) й використовує виключно підпрограми, послідовне виконання, розгалуження (вибір) і цикли (ітерації). Ця теорема є науковим положенням, використаним Е. Дейкстрою для обґрунтування його ідеї про використання в програмах лише трьох керуючих конструкцій: послідовного виконання, розгалужень і циклів. Теорема Бьома-Якопіні не розв'язує питання про те, чи слід застосовувати структурне програмування для розробки програмного забезпечення.

Деякі науковці використовували пуристській підхід до результату Бьома-Якопіні й стверджували, що навіть такі інструкції, як break і return в середини циклів, є поганим підходом до розробки алгоритмів і всі цикли повинні мати єдину точку виходу. Пряме застосування теореми Бьома-Якопіні може привести до додавання додаткових локальних змінних до структурованої програми, а також до деякого дублювання коду. Останнє питання в цьому контексті називають «проблемою циклу з половиною». 1973 року С. Рао Косараджу довів, що можливо уникати додавання додаткових змінних в структурне програмування, якщо допускаються багаторівневі виходи довільної глибини з циклів.

Контрольні питання

1. Дайте означення базової алгоритмічної конструкції.
2. Що таке структура слідування?
3. Які алгоритми називають лінійними?
4. Чи всі алгоритми, які містять структуру слідування лінійні? Обґрунтуйте.

5. Дайте означення структури розгалуження .Які бувають види розгалужень?

6. Що таке вкладене розгалуження?

7. Дайте означення структури вибору.

8. У чому полягає різниця між структурами “вибір” та “вибір-інакше”?

9. Дайте означення циклу. Назвіть види циклів.

10. Для яких видів циклів наперед відомою є кількість повторень?

11. Для яких видів циклів тіло циклу може не виконатись жодного разу?

12. Для яких видів циклів тіло циклу обов’язково виконається хоча б один раз?

13. Сформулюйте теорему Бьома-Якопіні.

Лекція 6. Рекурсія. Рекурсивні алгоритми

6.1 Поняття та використання рекурсії

Рекурсія – процес повторення чого-небудь самоподібним способом. Наприклад, вкладені віддзеркалення, вироблені двома точно паралельними один одному дзеркалами, є однією з форм нескінченної рекурсії. Даний термін має більш спеціальні значення в різних областях знань – від лінгвістики до логіки.

Рекурсія – це одна з фундаментальних концепцій у математиці й програмуванні. Рекурсія – це одна з форм мислення, цей потужний засіб, ще дозволяє будувати елегантні й виразні алгоритми.

Найбільш загальне застосування рекурсія знаходить у математиці й інформатиці. Тут вона є методом визначення функцій, при якому обумовлена функція застосована в тілі свого ж власного визначення. При цьому нескінченний набір випадків (значень функції) описується за допомогою кінцевого виразу, що для деяких випадків може посилатися на інші випадки, якщо при цьому не виникає циклів або нескінченного ланцюжка посилань. Фактично це спосіб визначення множини об'єктів через самого себе з використанням раніше заданих окремих правил.

Визначення, які використовують рекурсію, називаються *індуктивними*. Одним із прикладів подібного визначення є аксіоматична побудова множини натуральних чисел.

- Факторіал цілого додатного числа n (позначається $n!$) визначається як $n! = n \cdot (n - 1)!$ при $(n > 0)$ та $n! = 1$ при $(n = 0)$.
- Числа Фібоначі визначаються за допомогою рекурентного співвідношення:
  - Перше й друге числа Фібоначі рівні 1.
  - Для $n > 2$, $n - e$ число Фібоначі дорівнює сумі $(n - 1)$-го й $(n - 2)$-го чисел Фібоначі.
- Практично всі геометричні фрактали задаються у формі нескінченної рекурсії (наприклад, трикутник Серпінського).

– Рекурсивні акроніми: GNU (GNU Not Unix), PHP (PHP: Hypertext Preprocessor) і т.д.

Об'єкт називається *рекурсивним*, якщо він містить сам себе або визначений за допомогою самого себе.

Якщо процедура p містить явне звертання до самої себе, то вона називається *явно рекурсивною*. Якщо процедура p містить звертання до деякої процедури q, яка у свою чергу містить пряме або непряме звертання до p, то p – називається *побічно рекурсивною*.

Але рекурсивна програма не може викликати себе нескінченно, інакше вона ніколи не зупиниться, для цього у програмі (функції) повинен бути присутнім ще один важливий елемент – так звана термінальна умова, тобто умова за якої програма припиняє рекурсивний процес.

6.2 Рекурсія в програмуванні. Функції

У програмуванні рекурсія – виклик функції (процедури) з неї ж самої, безпосередньо (проста рекурсія) або через інші функції (складна рекурсія), наприклад, функція *A* викликає функцію *B*, а функція *B* – функцію *A*. Кількість вкладених викликів функції або процедури називається *глибиною рекурсії*.

Перевага рекурсивного визначення об'єкта полягає в тому, що таке скінченне визначення теоретично здатне описувати нескінченно велику кількість об'єктів. За допомогою рекурсивної програми ж можливо описати нескінченне обчислення, причому без явних повторень частин програми.

На практиці реалізація рекурсивних викликів функцій у мовах і середовищах програмування, як правило, опирається на механізм стеку викликів – адреси повернення й локальних змінних функцій записуються в стек, завдяки чому кожний наступний рекурсивний виклик цієї функції користується своїм набором локальних змінних і за рахунок цього працює коректно. Зворотним боком цього досить простого за структурою механізму є те, що на кожний рекурсивний виклик потрібна деяка кількість оперативної пам'яті комп'ютера, і при надмірно великій глибині рекурсії може наступити переповнення стека викликів. Внаслідок цього, зазвичай рекомендується уникати рекурсивних програм, які приводять (або при деяких умовах можуть приводити) до занадто великої глибини рекурсії.

Втім, є спеціальний тип рекурсії, що називається *«хвостовою рекурсією»*. Інтерпретатори й компілятори функціональних мов програмування, що підтримують оптимізацію коду (вихідного та/або що виконується), автоматично перетворять хвостову рекурсію до ітерації, завдяки чому забезпечується виконання алгоритмів із хвостовою рекурсією в обмеженому обсязі пам'яті. Такі рекурсивні обчислення, навіть якщо вони формально нескінченні (наприклад, коли за допомогою рекурсії організовується робота командного інтерпретатора, що приймає команди користувача), ніколи не приводять до вичерпання пам'яті. Однак, далеко не завжди стандарти мов програмування чітко визначають, яким саме умовам

повинна задовольняти рекурсивна функція, щоб транслятор гарантовано перетворив її в ітерацію. Один з рідкісних винятків – мова Scheme (діалект мови Lisp), опис якої містить всі необхідні відомості.

Будь-яку рекурсивну функцію можна замінити циклом і стеком.

6.3 Рекурентності

Рекурентність – це рекурсивне визначення функції. Вона широко поширена в математиці. Можливо, найбільш знайома Вам з такого роду функцій, як факторіал. Факторіал – це добуток натуральних чисел від одиниці до деякого даного натурального числа. Він визначається формулою:

$$N! = N((N-1)!, \quad \text{для } N >= 1 \text{ і } 0! = 1.$$

Це прямо відповідає нижченаведеному псевдокоду рекурсивної програми:

```
function int factorial(int N)  begin     if N = 0 then         factorial = 1
else
        factorial = N * factorial(N - 1)  end
```

Ця програма демонструє основні властивості рекурсивних програм: програма викликає сама себе (з меншим значенням аргументу), і в неї є термінальна умова, за якою вона прямо обчислює результат.

Необхідно також пам'ятати про те, що це – програма, а не формула: наприклад ні формула, ні програма не працюють із від'ємними *N*, але згубні наслідки спроби зробити обчислення для від'ємного числа більш помітні для програми, ніж для формули. Виклик factorial(-1) призведе до нескінченного рекурсивного циклу. Тому перед викликом даної програми потрібно робити перевірку умови незаперечності.

Друге добре відоме рекурентне співвідношення – співвідношення для визначення чисел Фібоначі. Числа Фібоначі – це елементи числової послідовності 1, 1, 2, 3, 5, 8, ..., у яких кожний наступний елемент дорівнює сумі попередніх.

$$F_N = F_{N-1} + F_{N-2}, \text{ де } N >= 2 \text{ і } F_0 = F_1 = 1.$$

І знову, рекурентність відповідає простій рекурсивній програмі, яку можна реалізувати на основі наступного псевдокоду:

```
function int fibonacci(int N)   begin     if N <= 1 then         fibonacci = 1
else
      fibonacci = fibonacci(N - 1) + fibonacci(N - 2)  end
```

Як ми побачимо, багато цікавих алгоритмів можна легко реалізувати за допомогою рекурсивних програм, і багато розроблювачів алгоритмів бажають реалізовувати алгоритми рекурсивно. Але часто трапляється також і так, що настільки ж цікавий алгоритм ховається в деталях нерекурсивної реалізації. Давайте розглянемо такий псевдокод алгоритму:

```
const max = 25
var int i; array of int F[0..max]

procedure fibonacci()
 begin     F[0] = 1     F[1] = 1
   for i = 2 to max do
      F[i] = F[i - 1] + F[i - 2]  end
```

Ця програма обчислює перші max чисел Фібоначі, використовуючи масив розміром max. Цей метод називається *ітераційним*.

Який же метод краще? Точних правил для вибору між рекурсивною й нерекурсивною версіями алгоритму рішення завдання не існує. Стислість і виразність більшості рекурсивних процедур спрощує їхнє читання й супровід. З іншого боку, виконання рекурсивних процедур вимагає великих витрат і пам'яті, і часу процесора в порівнянні з їхніми ітераційними аналогами.

Поділ навпіл

Велика частина алгоритмів використовує два рекурсивних виклики, кожний з яких працює приблизно з половиною вхідних даних. У дизайні алгоритмів таке явище називають "*поділ навпіл*"; його часто використовують для досягнення істотної економії.

Як приклад, давайте розглянемо завдання нанесення поділок на лінійку: на ній повинна бути мітка в точці 1/2", мітка трохи коротша через кожні 1/4", ще більш коротка через 1/8" і так далі (рис. 6.1).

Рисунок 6.1 – Лінійка

Ми також припускаємо, що в нас є процедура mark(*x*, *h*) для нанесення мітки висотою *h* одиниць на лінійку в позицію *x*. Центральна мітка повинна мати висоту n одиниць, мітки в центрах лівої й правої половинок – висоту (*n* – 1) одиниць, і так далі. Наступний псевдокод рекурсивної програми ілюструє прямий шлях досягнення нашої мети:

```
procedure rule(int l, int r, int h)  var int m  begin     if h > 0 then     begin
      m = (l + r) div 2       mark( m, h )       rule( l, m, h - 1)       rule( m,
r, h - 1)     end  end
```

Ідея цього методу полягає в наступному: для того, щоб промаркувати лінійку, ми спершу наносимо довгу мітку в її середині. Це ділить її на дві рівні частини. Тепер ми наносимо (більш короткі) мітки в середині кожної із цих половинок, використовуючи ту ж саму процедуру.

Необхідно звертати особливу увагу на термінальну умову рекурсивної програми – у протилежному випадку вона ніколи не зупиниться! У вищенаведеній програмі перевіряємо, коли ми повинні нанести мітку висотою 0. Розглянемо приклад, як результат виклику rule(0, 8, 3). Ми ставимо мітку по середині й викликаємо rule для лівої половини, потім робимо теж саме для лівої половини, і так далі поки висота мітки не стане дорівнювати 0. В остаточному підсумку ми вертаємося до rule і розмічаємо праву половину подібним чином.

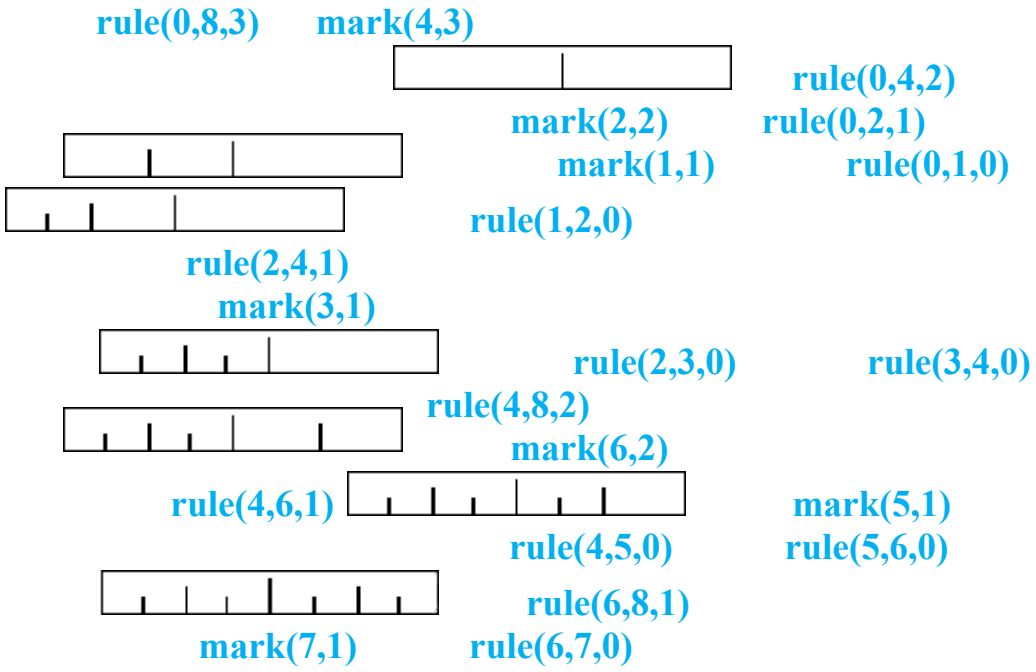


Рисунок 6.2 – Ілюстрація послідовності рекурсивних викликів

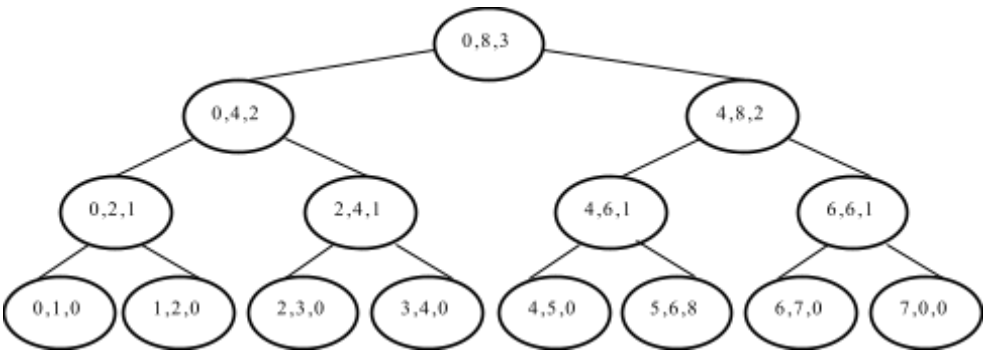


Рисунок 6.3 – Дерево рекурсивних викликів для малювання лінійки

Інший нерекурсивний алгоритм полягає в тому, щоб малювати спершу найкоротші мітки, потім трохи менш короткі й так далі, як показано в наступному, досить компактному, нерекурсивному псевдокоді:

```
procedure draw(int l, int r, int h)  var  int i, int j  begin      j = 1      for i =
1 to h do       begin
                for x = 0 to (l + r) div j do          mark(l + j + x * (j + j), i)          j =
j + j      end  end
```

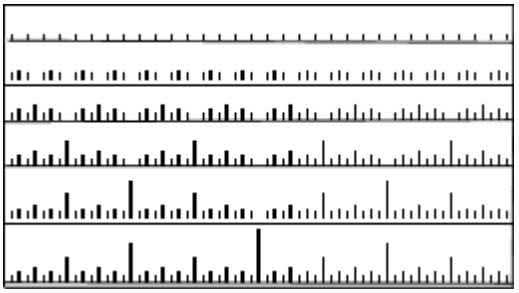


Рисунок 6.4 – Нерекурсивне малювання лінійки

Зараз створимо двомірний візерунок, що демонструє як проста рекурсія може дати рішення тому, що здається складним.

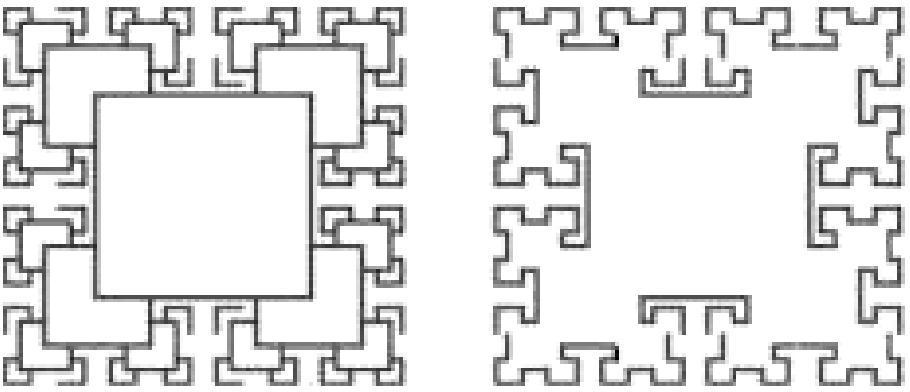


Рисунок 6.5 – Фрактальна зірка (ліворуч) і її обриси (праворуч)

Приклад псевдокоду алгоритму створення фрактальної зірки:

```
var int r

 procedure star(int x, int y, int r)   begin
    if r > 2 then begin          star(x + r, y + r, r div 2)          star(x + r, y - r, r
div 2)          star(x - r, y - r, r div 2)          star(x - r, y + r, r div 2)
              drawFilledRrectangle(x - r, y - r, x + r, y + r)      end   end   begin

 ...  // тут повинне бути очищення екрану

    write('Enter the length of star: ')      read(r)

 ...  // тут повинні бути функції налаштування графіки   do
    star(getMaxX() div 2, getMaxY() div 2, r)      repeat
```

```
until (not keyPressedEsc)  end
```

Примітив для малювання тут – просто процедура, яка малює квадрат розміром $2r$ із центром в (x, y).

Рекурсивно визначені геометричні візерунки подібні цьому називають *фракталами*. Якщо використовується більш складний примітив для малювання й більш складні рекурсивні виклики (особливо на дійсній і комплексній площинах), то можна розробити візерунки вражаючої краси й складності.

Відомим прикладом фракталу є трикутник Серпінського (рис. 6.6).

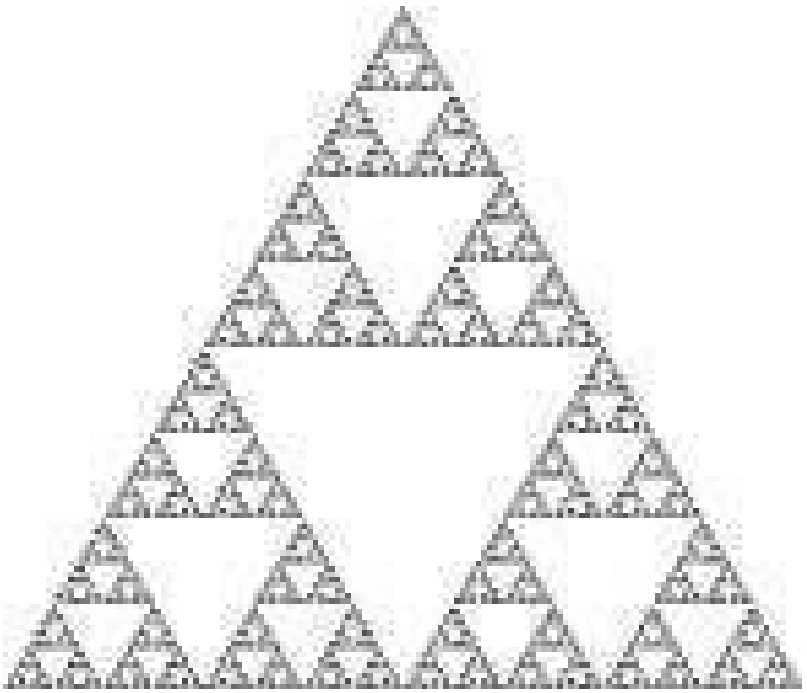


Рисунок 6.6 – Трикутник Серпінського

6.4 Рекурсивні структури даних

Опис типу даних може містити посилання на самого себе. Подібні структури використовуються при описі списків і графів. Спосіб створення рекурсивних типів даних дуже залежить від засобів тієї чи іншої мови програмування і, зокрема, реалізації у ній вказівників. Наведемо приклад створення рекурсивної структури даних не на псевдокоді, а на коді реальної мови програмування.

Приклад 6.1. Опис списку на мові C++

```
class element_of_list
{
  element_of_list *next; /* посилання на наступний елемент того ж типу */   int data; /* деякі дані */
};
```

Рекурсивна структура даних найчастіше спричиняє застосування рекурсії для обробки цих даних.

Корекурсія

Корекурсія – у теорії категорій та інформатиці тип операції, дуальний до рекурсії. Зазвичай корекурсія використовується (разом з механізмом ледачих обчислень) для генерації нескінченних структур даних.

Загальні зауваження

Правило використання корекурсії на коданих (прикладом яких є потоки) дуальне правилу застосування рекурсії на даних. Замість *згортання* структури даних виходячи з початкового значення аргументу, корекурсія *розгортає* результат на основі початкового значення аргументу. Необхідно відзначити, що корекурсія *створює* потенційно нескінченні структури даних, у той час як звичайна рекурсія *аналізує (розбирає)* по необхідності скінченні структури даних. Звичайна рекурсія незастосовна до коданих, оскільки процес аналізу може ніколи не зупинитися. Відповідно, корекурсія не може продукувати дані, оскільки дані завжди скінченні.

Приклади корекурсії

Приклад використання механізму корекурсії мовою Haskell (обчислення нескінченного списку чисел Фібоначі):

```
fibs = 0:1: zipWith (+) fibs(tail fibs)
```

Інший приклад – обчислення нескінченного списку простих чисел:

```
primes = eratosthenes [2..]    where
  eratosthenes (x:xs) = x:eratosthenes (filter ((/=
0).('mod' x)) xs)
```

Дана функція реалізує алгоритм «решето Ератосфена», причому робить це самим безпосереднім чином.

Наведені приклади мовою Haskell не зовсім коректні, оскільки в мові немає ідіоми коданих. У зазначених прикладах кодані тільки емулюються за допомогою нескінченного списку.

Контрольні питання

1. В чому різниця між явно і побічно рекурсивними процедурами?
2. Який важливий елемент повинен бути присутнім у програмі (функції) для її коректної роботи?
3. Які широко поширені рекурентності Ви знаєте з математики?
4. Чому перед викликом програми для обчислення факторіалу потрібно робити перевірку умови незаперечності?

5. Які переваги й недоліки використання рекурсивних процедур?

6. На Ваш погляд, який із наведених вище прикладів для малювання лінійки (а саме рекурсивний розподіл навпіл чи нерекурсивний алгоритм) кращий і чому?

7. Дайте означення фракталу.

8. Дайте означення глибини рекурсії.

9. Чому рекомендують уникати рекурсивних програм?

10. Опишіть порядок роботи “хвостової рекурсії”.

11. Дайте визначення корекурсії.

Лекція 7 – 8 . Алгоритми циклічної структури.

7.1 Алгоритми циклічної структури

Цикл – це повторення одних і тих самих дій (кроків). Послідовність дій, які повторюються в циклі, називається *тілом циклу*. Існує кілька типів *алгоритмів циклічної структури*. На рисунку 7.1 показаний цикл з передумовою, а на рис. 7.2 показаний цикл з постумовою, які називаються *умовними циклічними алгоритмами*.

- у циклі передумови умова перевіряється перед тілом петлі, у циклі після умови – після тіла петлі;
- у циклі післяумови тіло циклу виконується хоча б один раз, у циклі з передумовою тіло циклу може взагалі не виконуватися;
- у циклі з передумовою перевіряється умова продовження циклу, в циклі з постумовою перевіряється умова виходу з циклу.

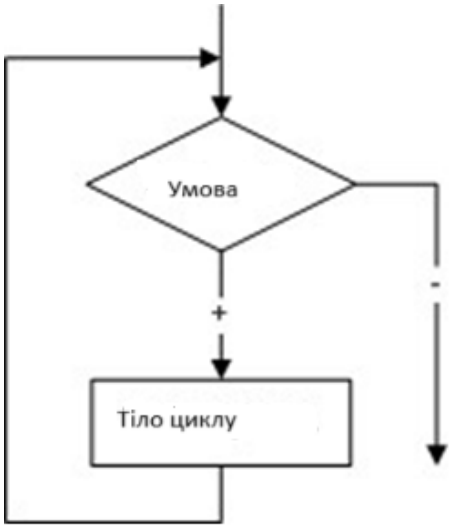


Рис. 7.1 Алгоритм циклічної структури

Рис 7.2 Алгоритм циклічної структури з післяумовою

з передумовою

При написанні умовних циклічних алгоритмів слід мати на увазі наступне.

По-перше, щоб цикл мав шанс коли-небудь закінчитися, вміст його тіла обов'язково має впливати на стан циклу.

По-друге, умова повинна складатися з допустимих виразів і значень, визначених перед першим виконанням тіла циклу.

Крім того, існує так званий безумовний циклічний алгоритм (рис. 7.3), який стане в нагоді, якщо ви знаєте, скільки разів потрібно виконати тіло циклу.

Виконання безумовного циклічного алгоритму починається з присвоєння змінної **i** початкового значення **in**. Потім перевіряється, чи не перевищує змінна **i** кінцеве значення **ik**. Якщо це відбувається, цикл вважається завершеним, і управління передається оператору**,** що слідує за тілом цикле, змінює своє значення відповідно до заданого кроку **di**. Далі знову перевіряється значення змінної **i** та алгоритм повторюється.

Зрозуміло, що безумовний циклічний алгоритм може бути замінений будь-яким умовним. Наприклад рис. 7.4.

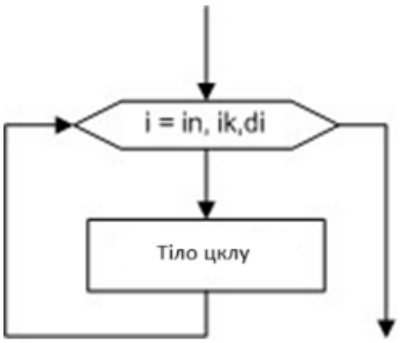


Рис. 7.3. Алгоритм циклічної структури без умови

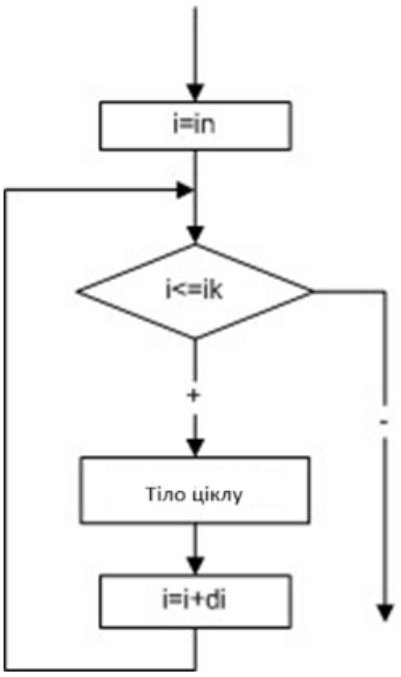


Рис. 7.4. Умовний циклічний алгоритм з відомою кількістю повторень

Зауважимо, що змінна **i** називається *параметром циклу*, оскільки це змінна, яка змінюється всередині циклу за певним законом і впливає на його закінчення.

Розглянемо використання алгоритмів циклічної структури на конкретних прикладах.

ПРИКЛАД 7.1. Знайдіть найбільший спільний дільник (НСД) двох натуральних чисел A і B.

Вхідні дані**:** `A` і `B`. Вихід**:** `A` – НСД.

Для вирішення цієї задачі скористаємося алгоритмом Евкліда:

Будемо зменшувати кожен раз більше з чисел на величину меншого до тих пір, поки обидва значення не стануть рівними, як показано в таблиці 7.1.

Таблиця 7.1. Пошук НСД для чисел A=25 та B=15.

Вхідні данні	**Перший крок**	**Другий крок**	**Третій крок**	**НСД(A,B)=5**
A=25	A=10	A=10	A=5	
B=15	B=15	B=5	B=5	

У блок-схемі вирішення проблеми, представленої на рис. 7.5, для вирішення задачі використовується цикл з передумовою, тобто тіло циклу повторюється до тих пір, поки A не стане рівним B.

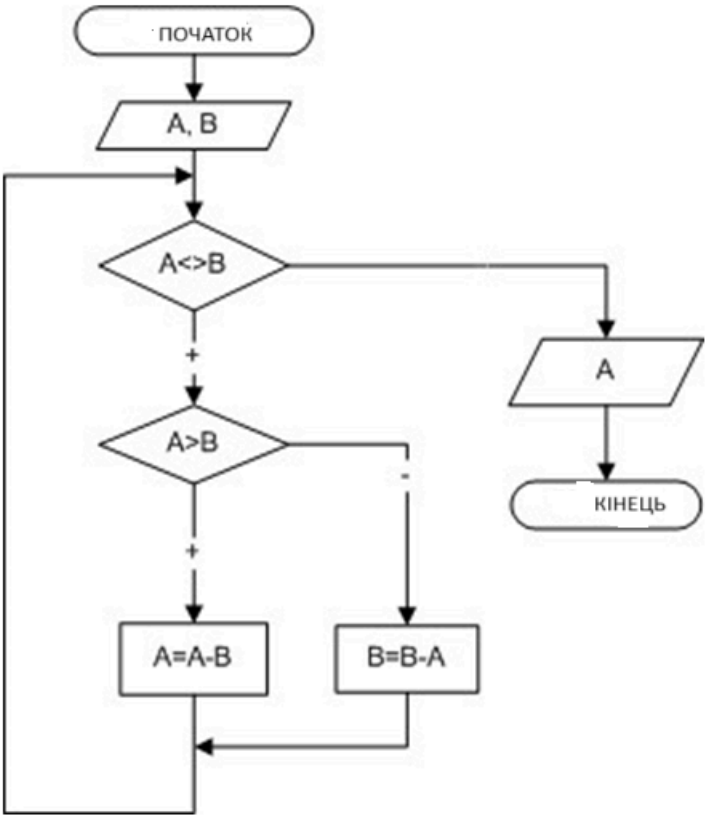


Рис. 7.5. Пошук найбільшого спільного дільнику двох чисел

ПРИКЛАД 7.2. Вводиться послідовність чисел, 0 - кінець послідовності. Визначте, чи містить послідовність принаймні два рівних сусідніх числа.

Вхідні дані**:** `X0` - поточний член послідовності, `X1` - наступний член послідовності.

Вихід**:** повідомлення про те, що в послідовності є два рівноправних сусіда**.**

Допоміжні змінні: `Fl` - логічна змінна, яка зберігає значення "true", якщо в послідовності є рівні сусідні доданки і "false" - в іншому випадку.

Блок-схема вирішення проблеми показана на рис. 7.6. Використання циклу з постумовою тут обґрунтовується тим, що необхідно спочатку порівняти два елементи послідовності, а потім прийняти рішення про завершення циклу.

8.1 Цикли з відомим числом повторень

У наведених вище прикладах умова задачі таке, що невідомо, скільки разів тіло циклу буде повторюватися. Такі цикли називаються *циклами з невідомою кількістю повторень* (взагалі невідомо, чи закінчиться цикл взагалі). Цикл, число повторень якого відомо заздалегідь або може бути

визначено за вихідними даними, називається *циклом з відомим числом повторень*.

Давайте розглянемо кілька прикладів з використанням таких петель.

ПРИКЛАД 7.3. Скласти таблицю значень функції $y = esin(x)cos(x)$ на відрізку [0;π] з кроком 0,1.

Вхідні дані**:** початок аргументу дорівнює 0, кінець аргументу — π, а крок зміни аргументу — 0,1.

Висновок**:** множина значень аргументу `X` та відповідний набір значень функції `Y`**.**

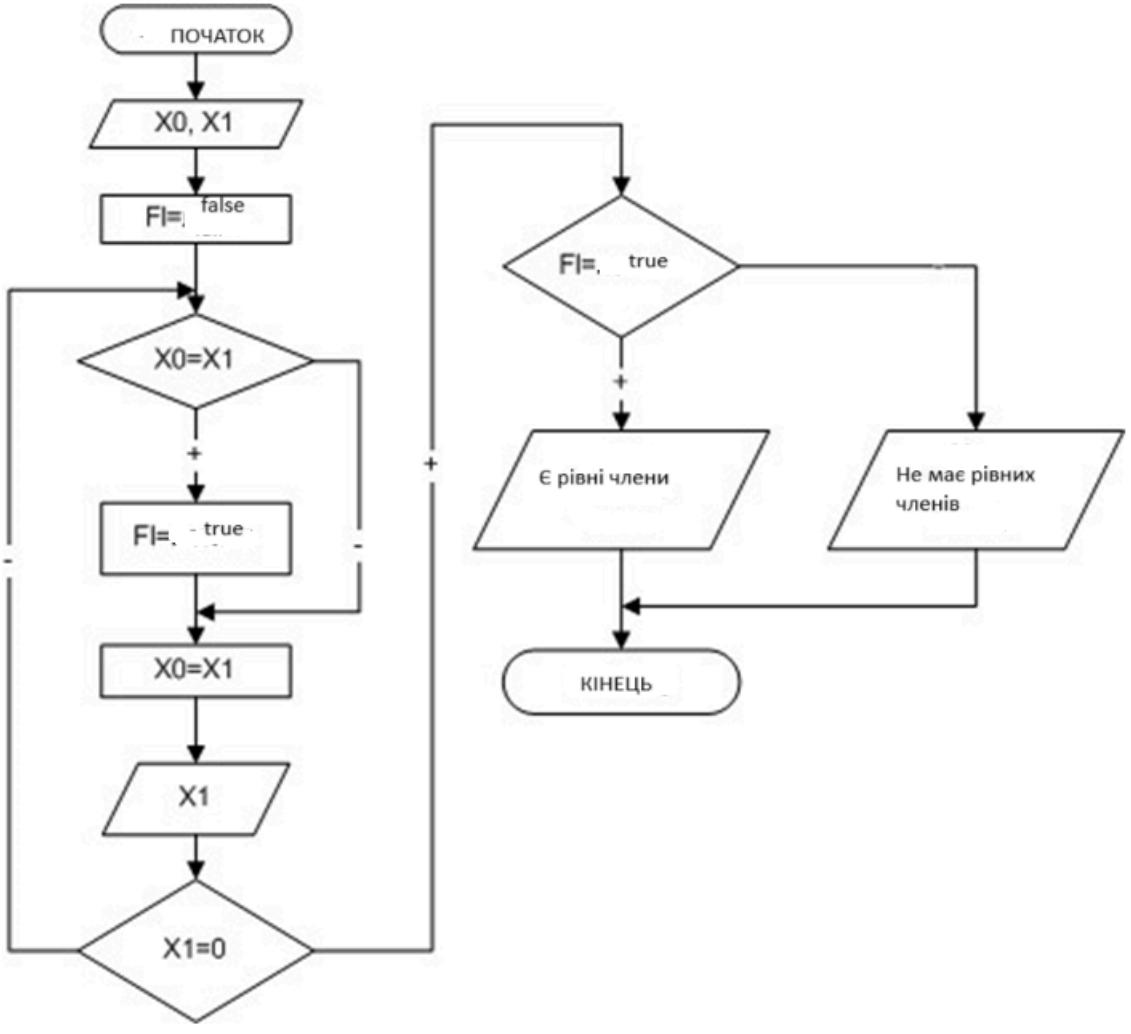


Рис. 7.6. Знаходження рівних сусідів послідовності

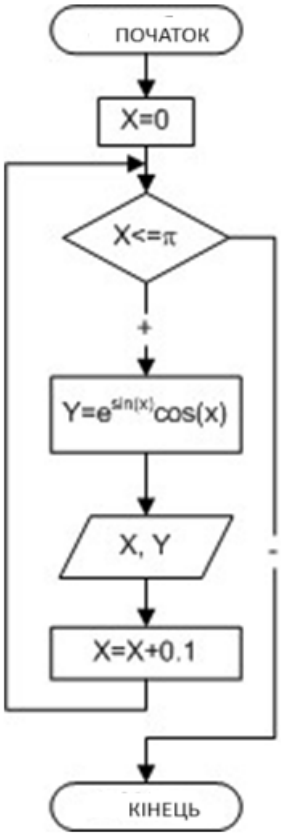


Рис. 7.7. Створення таблиці значень функції *y = esin(x)cos(x)*(1–й спосіб)

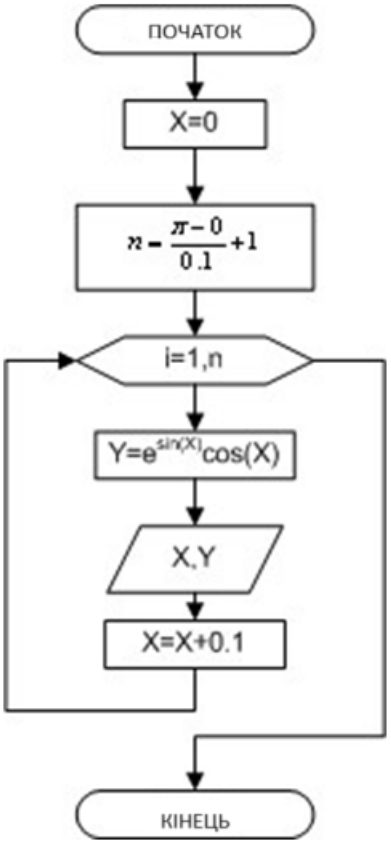


Рис. 7.8. Створення таблиці значень функції *y = esin(x)cos(x)*(2–й спосіб)

Кількість повторень циклу явно не вказується в операторі задачі, тому її можна вирішити за допомогою циклу з передумовою (рис. 7.7). З іншого боку, відомо, як змінюється параметр циклу `X` і які його початкове і кінцеве значення, отже, попередньо визначивши число **n** повторень тіла циклу, як показано на рис. 7.8, можна використовувати оператор безумовного циклу. Так, якщо параметр циклу `x` приймає значення в діапазоні від `xn` до `xk`, що змінюються з кроком `dx`, то число повторень тіла циклу можна визначити за формулою:

$$n = \frac{x_k - x_n}{dx} + 1 \qquad (7.1)$$

округлення результату ділення до цілого числа.

ПРИКЛАД 7.4. Обчисліть факторіал числа N (N!=1·2·3 ... · H).

Вхідні дані: `N` — ціле число, факторіал якого потрібно обчислити. Вихідні дані: `factorial` — факторіальне значення числа `N`, добуток чисел від 1 до `N`, ціле число.

Проміжні дані: `i` - цілочисельна змінна, яка приймає значення від 2 до `N` з кроком в 1, параметр циклу. Блок-схема показана на рис. 7.9.

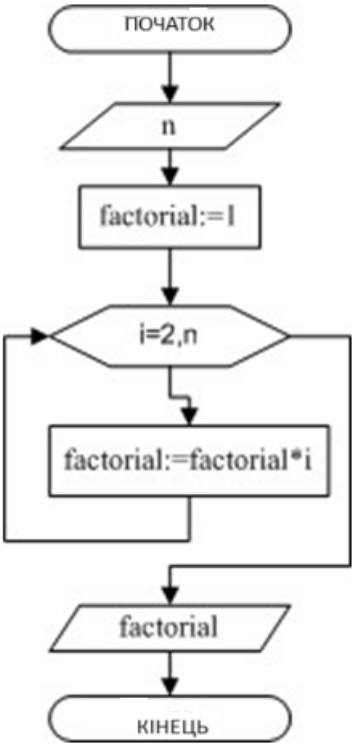


Рис. 7.9. Факторіальний розрахунок

Отже, вводиться число **N**. Змінній **`factorial`,** яка призначена для зберігання значення добутку послідовності чисел, присвоюється початкове значення, рівне одиниці. Потім організовується цикл, параметром якого є змінна **`i`**. Якщо значення параметру циклу менше або дорівнює **N**, то виконується оператор тіла циклу, в якому з ділянки пам'яті зберігається **`factorial`**. Попереднє значення добутку зчитується, множиться на поточне значення параметра циклу, а результат поміщається назад у місце пам'яті під назвою **`factorial`.** Коли параметр **i** стає більшим за **N,** цикл закінчується і значення виводиться змінною **`factorial`**, яка була обчислена в тілі циклу.

ПРИКЛАД 7.5. Обчисліть $\mathbf{a^n}$ ***(n>0)***.

Вхідні дані: **`a`** - дійсне число, яке потрібно піднести до додатного цілого ступеня **`n`**.

Вихід: **`p`** (дійсне число) **є** результатом піднесення дійсного числа **`a`** до додатного цілого ступеня **`n`**.

Проміжні дані: **i**— цілочисельна змінна, яка приймає значення від **1 до n** з кроком **1**, параметр циклу.

Блок-схема показана на рис. 7.10.

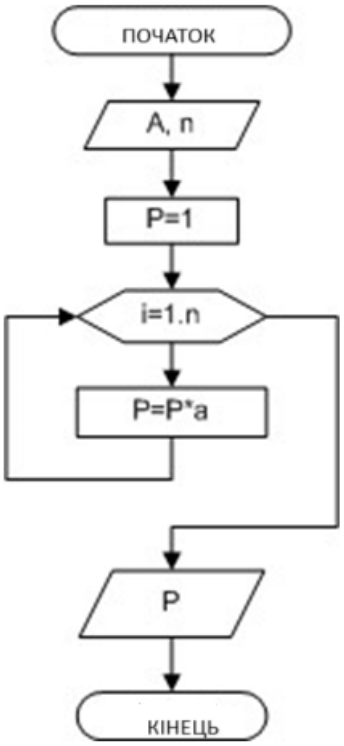


Рис. 7.10. Піднесення дійсного числа до цілого числа

Відомо, що для отримання цілого ступеня n числа a потрібно помножити його на себе n разів. Результат цього множення буде зберігатися в ділянці пам'яті з іменем p. При виконанні чергового циклу з цього розділу попереднє значення буде прочитано, помножене на a і знову записано в пам'ять p. Цикл виконується n разів.

У таблиці 7.2 показаний протокол виконання алгоритму при піднесенні числа 2 в п'ятий ступінь: a=2, n=5. Такі таблиці, що заповнюються вручну, використовуються для тестування - перевірки всіх етапів програми.

Таблиця 7.2. Процес піднесення a до ступеня n

i		1	2	3	4	5
P	1	2	4	8	16	32

ПРИКЛАД 7.6. Обчислити суму натуральних парних чисел, що не перевищують N**.**

Вхідні дані**: N** є цілим числом.

Висновок**: S** – сума парних чисел**.**

Проміжні дані**: i** - змінна, яка приймає значення від 2 до N з кроком 2, а тому також має ціле значення.

При додаванні декількох чисел необхідно накопичувати результат в певній області пам'яті, кожен раз зчитуючи попереднє значення суми з цієї області і додаючи до нього наступний доданок. Для виконання першого оператора накопичення суми з області пам'яті необхідно взяти число, яке б не впливало на результат додавання. Тобто перед початком циклу змінна, призначена для накопичення суми, повинна бути встановлена на нуль. Блок-схема вирішення цієї задачі представлена на рис. 7.11.

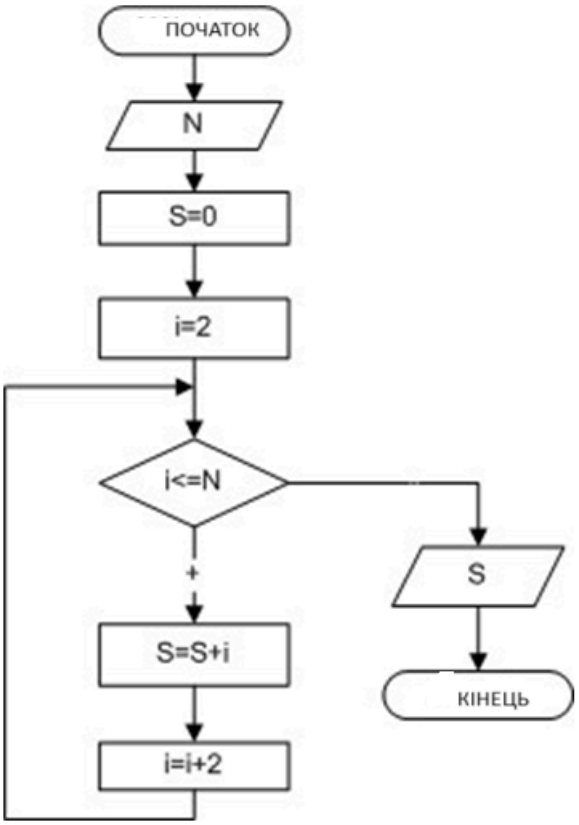


Рис. 7.11. Обчислення суми парних, натуральних чисел

У таблиці 7.3 наведені результати тестування алгоритму для **n=7**. Неважко помітити, що при непарних значеннях параметра циклу значення змінної, призначеної для накопичення суми, не змінюється.

Таблиця 7.3. Підсумовування парних чисел

i		1	2	3	4	5	6	7
S	0	0	2	2	6	6	12	12

ПРИКЛАД 7.7. Дано натуральне число N. Визначте *K* як кількість дільників цього числа, які не перевищують його (*N=12, його дільники 1, 2, 3, 4, 6, K=5).*

Вхідні дані**:** N є цілим числом.

Висновок**:** Ціле `число K` – це кількість дільників `N`**.**

Проміжні дані**:** `i` - параметр циклу, можливі дільники `числа N`.

У блок-схемі, наведеній на рис.7.12, реалізований наступний алгоритм: у змінну K, яка призначена для підрахунку кількості дільників заданого числа ставиться значення, яке б не впливало на результат, тобто нуль `N`; а значення змінної `K` слід збільшити на одиницю. Петлю слід повторити `N/2` разів.

У таблиці 7.4 наведені результати тестування алгоритму при визначенні дільників числа `N = 12`.

Таблиця 7.4. Визначення дільників числа `N`

i		1	2	3	4	5	6
K	0	1	2	3	4	4	5

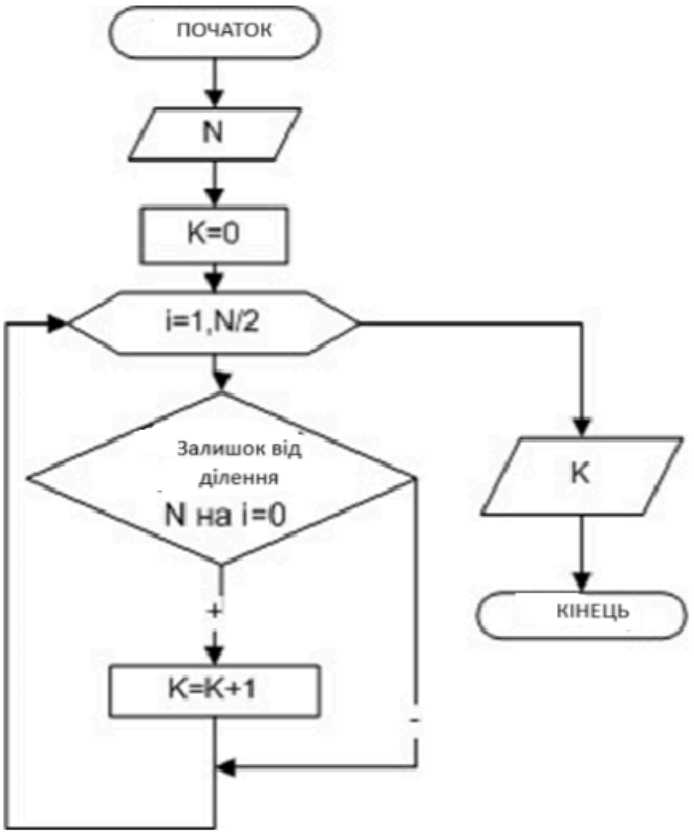


Рис. 7.12. Визначення числа дільників натурального числа

ПРИКЛАД 7.8. Дано натуральне число N. Натуральне число N називається *простим*, якщо воно рівномірно ділиться тільки на одиницю і N. Число 13 є простим, тому що воно ділиться тільки на 1 і 13, *N=12 не є простим, так як ділиться на 1, 2, 3, 4, 6 і 12***.**

Вхідні дані**:** `N` є цілим числом. Вихідні дані**:** Повідомлення**.**

Проміжні дані**:** `i` - параметр циклу, можливі дільники числа `N`.

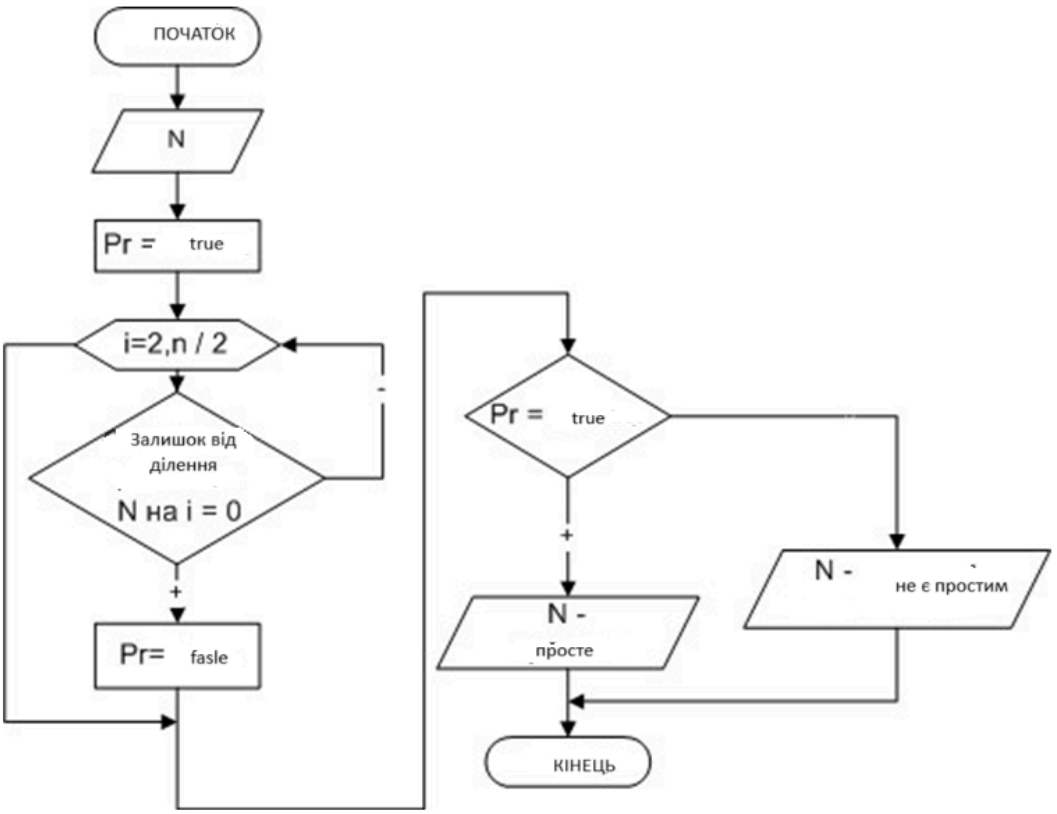


Рис. 7.13. Визначення простого числа

Алгоритм вирішення цієї задачі (рис. 7.13) полягає в тому, що число `N` ділиться на параметр циклу `i`, який змінюється в діапазоні від 2 до `N/2`. Якщо серед значень параметра, що ділить дане число на ціле число, немає числа, то `N` є простим числом, в іншому випадку воно не є простим числом. Зауважимо, що алгоритм передбачає два виходи з циклу. коли всі значення параметрів вичерпані, а друге - рано. Немає сенсу продовжувати цикл, якщо знайдено хоча б один дільник із заданої області модифікації параметра.

ПРИКЛАД 7.9. Визначте кількість простих чисел у інтервалі від *N* до *M*, де *N* і *M* – натуральні числа.

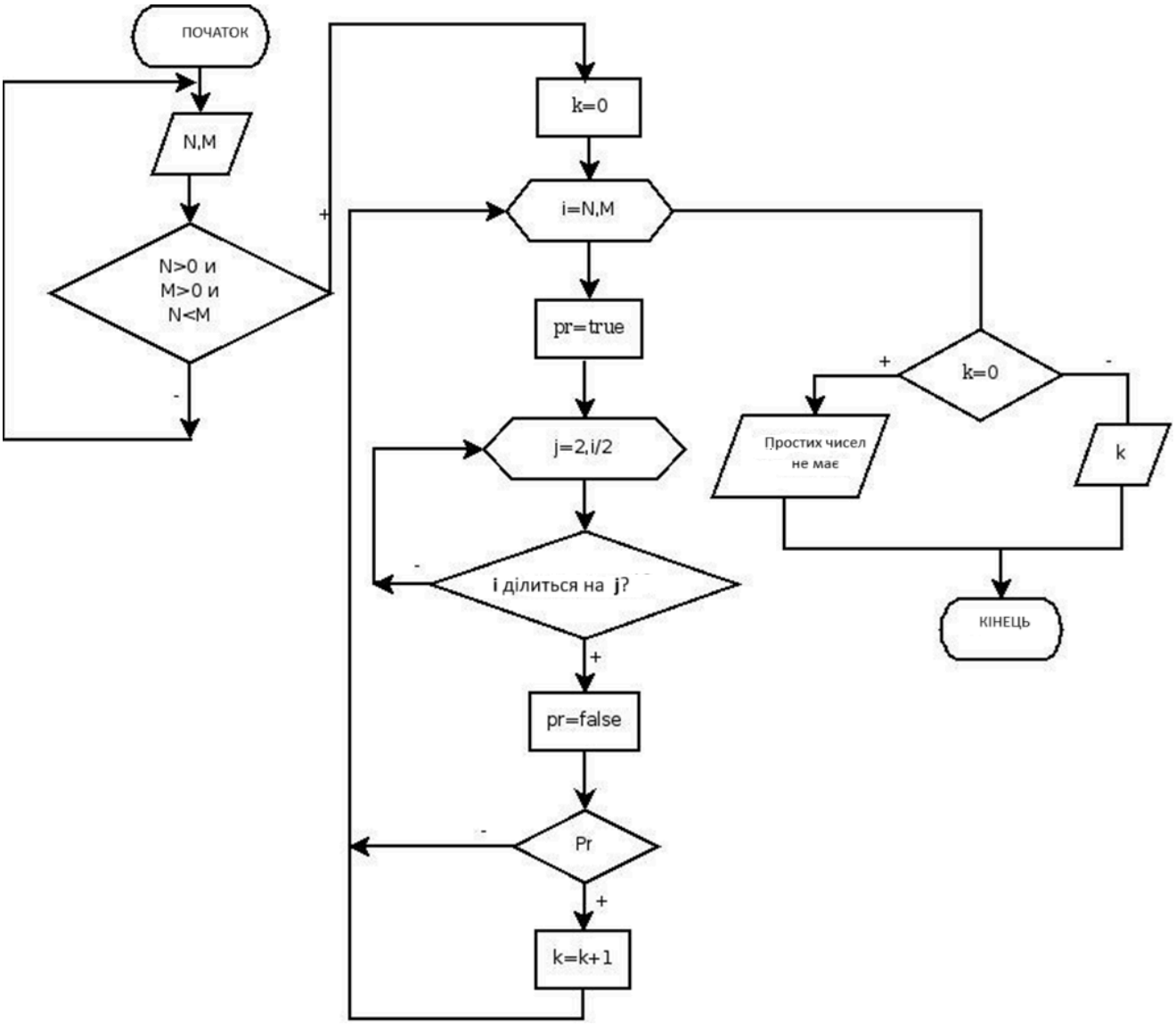


Рис. 7.14 Знайти прості числа в заданому інтервалі

ПРИКЛАД 7.10. Обчисліть значення **y**, що відповідають кожному значенню **x** (xn≤x≤xk, dx) по формулі:

y= e^{-x}sin(x) ,|x|≤a

e^{-x^2} cos(x),|x|>a

Знайти максимальне і мінімальне значення y.

Входи: xn, xk, dx, a.

Висновок*:* набір значень y, y_{max} — максимальне значення y, y_{min} — мінімальне значення y.

Алгоритм знаходження мінімального (максимального) значення змінної y можна описати наступним чином. Нехай мінімальне (максимальне) значення y зберігається в змінній y_{min} (y_{max}). Перед початком циклу цій змінній присвоюється дуже велике (мало) значення, наприклад, 10^{20} (-10^{20}). Потім в циклі порівнюються значення y і y_{min} (y_{max}), і якщо є значення **y**,

яке менше (більше), ніж зберігається в y_{min} (y_{max}), то його потрібно записати в змінну y_{min} (y_{max}). Таким чином, всі значення **y** неявно порівнюються між собою, і в кінці циклу в змінній y_{min} (y_{max}) збереже найменше (найбільше) серед значень **y**.

Блок-схема алгоритму зображено на рис. 7.15.

ПРИКЛАД 7.11. Обчисліть значення **z**, які відповідають кожному значенню

x ($x = 1$; $dx = 0.5$) по формулі: $z = \ln ln\, x * \sqrt{(\frac{x}{x^3+1})}$

Рахуйте z, доки вираз кореня не стане більшим або рівним 0,02. Визначте k - це число z, що обчислюється.

Вхідні дані: x, dx.

Висновок: множина значень z , k – кількість обчислених z.

Тут умова кінця циклу явно вказана в прикладі умови. Цикл закінчиться, коли кореневий вираз буде менше 0,02. Блок-схема алгоритму зображена на рис. 7.16.

ПРИКЛАД 7.12. Обчислити від'ємний корінь рівняння $x^3-x+0.5=0$, за допомогою повторюваної формули: $x_k + 1 = \sqrt[3]{x_k - 0.5}$ $(k=0,1,2,...)$. $x_0= -1.3;$ $\varepsilon=10^{-4}$. Визначити кількість ітерацій (кроків циклу); Зупиніть обчислення, коли умову виконано $|x_{k+1}-x_k|< \varepsilon$.

Вхідні дані: x_0, E (точність ε).

Вихідні дані: k — кількість ітерацій; x — корінь.

Проміжні дані: x_1 — подальше значення x_k.

В даному прикладі обчислюється набір значень $x_1, x_2, x_3,...X_K$. Скільки їх є, заздалегідь невідомо, але кінцеве значення змінної x_k і є коренем рівняння. У кожному циклі необхідно порівнювати поточні і попередні значення. Тому для розрахунку можна використовувати тільки дві змінні: x_0 - це попереднє значення і x_1 - поточне значення. Далі необхідно організувати цикл, який закінчиться, якщо модуль різниці між поточним і попереднім значеннями стане менше заданої точності. У тілі циклу на кожному новому кроці необхідно записувати поточне значення в змінну x_0 і перераховувати x_1 за формулою на рис. 7.17.

ПРИКЛАД 7.13. Обчислення значень функцій $y = chx = \sum_{k=1}^{\infty} U_k$, використовуючи рекурентну формулу $U_{k+1}=\frac{x^2}{(2k-1)2k}U_k$ $(k = 1,2,3,...)$.

Зупиніть обчислення, коли виконується умова $U_{k+1}|<\varepsilon$. Визначте кількість ітерацій. Початковими значеннями параметрів є $U_1 = 1;$ $x = 5.5$; точність ε прийняти рівної 10^{-4}.

Вхідні дані: U – початкове наближення, x, E – точність (ε).

Вихідні дані: y — значення обчислюваної функції, k — кількість ітерацій. Блок-схема алгоритму зображена на рис. 7.18.

На відміну від попереднього прикладу, тут достатньо однієї змінної U, яка буде зберігати поточне значення y. На кожному кроці циклічного алгоритму потрібно перераховувати значення y і додавати до нього U.

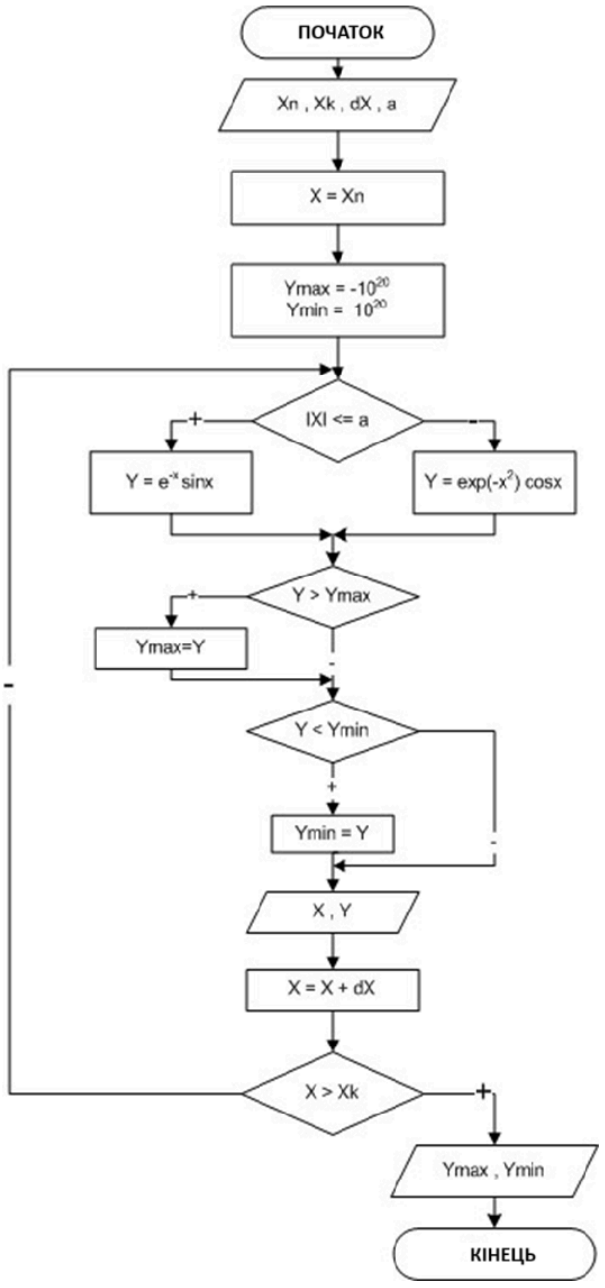


Рис. 7.15. Блок-схема алгоритму прикладу 7.10

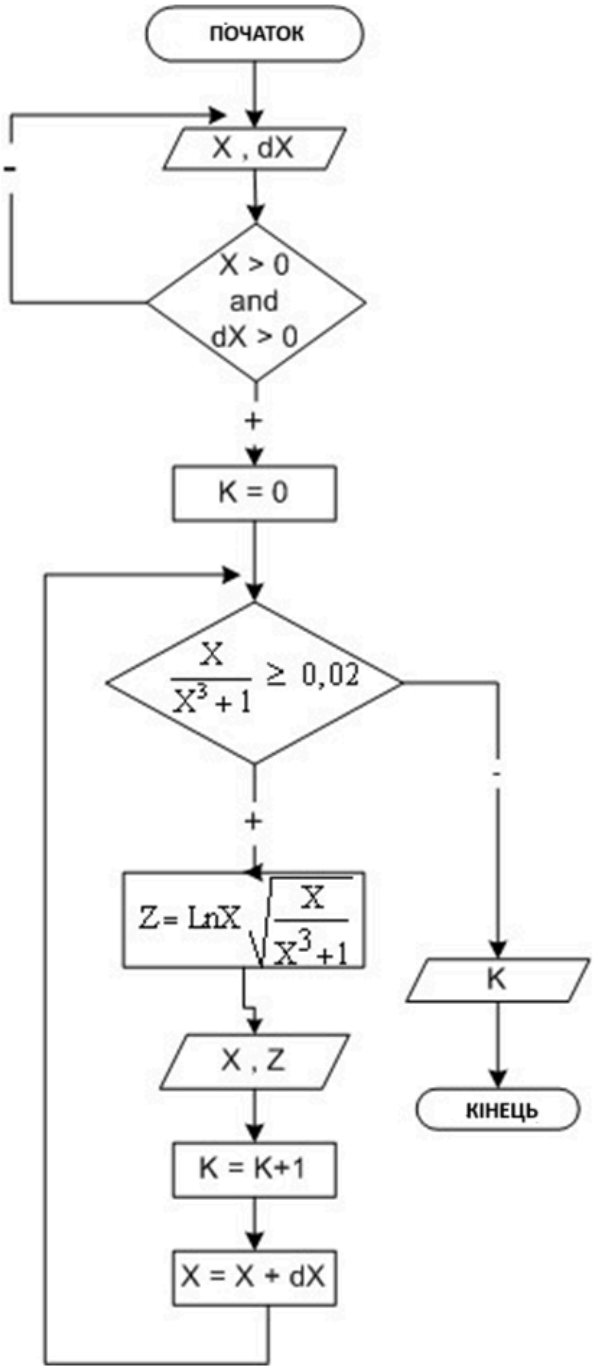


Рис. 7.16. Блок-схема алгоритму прикладу 7.11

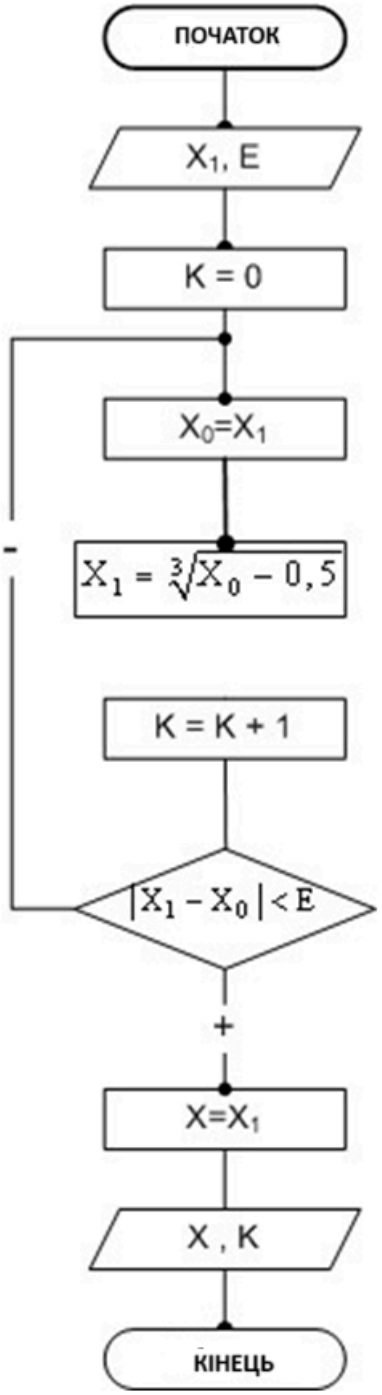


Рис. 7.17. Блок-схема алгоритму прикладу 7.12

ПРИКЛАД 7.14. Вводиться послідовність цілих чисел, 0 - кінець послідовності. Знайдіть серед позитивних мінімум, якщо таких значень кілька, визначте, скільки їх.

Блок-схема вирішення поставленої задачі наведена на рис. 7.19.

Для вирішення цієї задачі використовуємо логічну змінну `Pr`, яка визначає наявність позитивних чисел у послідовності (`Pr=false`, якщо немає позитивних чисел). Основний цикл працює до тих пір, поки N≠0. Кожен раз, коли ми входимо в цикл, перевіряємо знак `N`. Якщо `N>0` і

Pr=false, то це вказує на те, що це перше позитивне число. У цьому випадку ми оголошуємо це число мінімальним, кількість мінімальних чисел (k) дорівнює 1, а атрибут Pr =true. Якщо N>0 і Pr ≠ false, то порівнюємо поточне значення N з min. Якщо N<min, то це N оголошується мінімальним, число мінімальних чисел (k) дорівнює 1, якщо N = min, то число мінімальних значень збільшується на 1 (k=k+1). Останньою дією в циклі є введення наступного значення N.

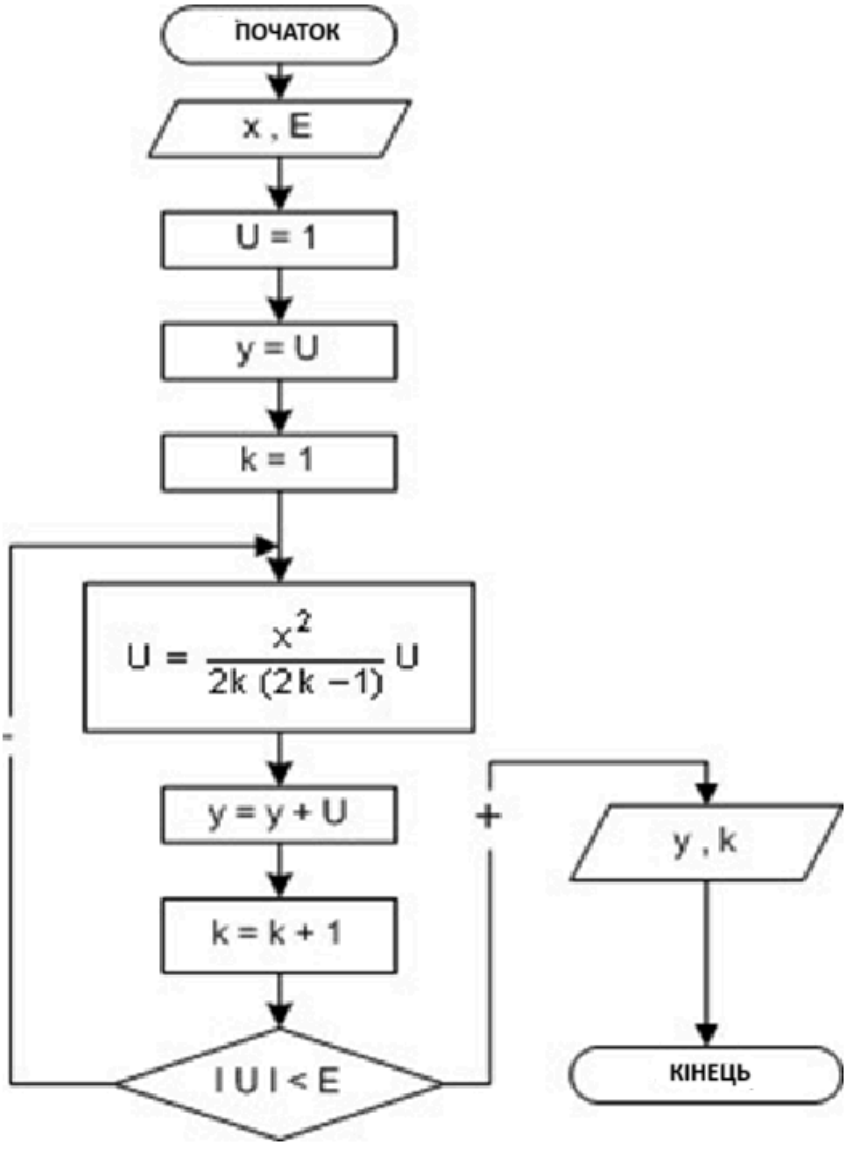


Рис. 7.18. Блок-схема алгоритму прикладу 7.13

ПРИКЛАД 7.15. Дано натуральне число N. Визначте найбільшу цифру і її положення в числі (N = 573863, найбільше - число 8, його позиція - четверте зліва).

Вхідні дані**:** N – ціле число.

Вихідні дані**:** `max` – значення найбільшої цифри в числі, `pos` – положення цієї цифри в числі**.**
Проміжні дані**:** `i` – параметр циклу, `kol` – кількість цифр у числі, `M` – змінна для тимчасового зберігання значення `N`**.**

Розіб'ємо розв'язання цієї задачі на два етапи. Спочатку знайдемо кількість цифр в даному числі (рис. 7.20), а потім визначимо найбільшу цифру і її положення (рис. 7.21). Для того щоб обчислити кількість цифр в числі, нам потрібно визначити, скільки разів дане число можна поділити на десять цілком. Наприклад, припустимо, `N = 12345`, тоді кількість цифр `kol` = 5. Результати проведених розрахунків зведені в таблицю 7.5.

Таблиця 7.5. Визначення кількості цифр числа

kol	N
1	12345
2	12345 div10=1234
3	1234 div 10=123
4	123 div 10=12
5	12 div 10=1
	1 div 10=0

Процес визначення поточного розряду числа N=12345 представлений в таблиці 7.6.

Таблиця 7.6. Визначення поточної цифри числа

i	Число M	Цифра
1	12345	12345 mod 10 = 5
2	12345 div10=1234	1234 mod 10 = 4
3	1234 div 10=123	123 mod 10 = 3
4	123 div 10=12	12 mod 10 = 2
5	12 div 10=1	1 mod 10 = 1

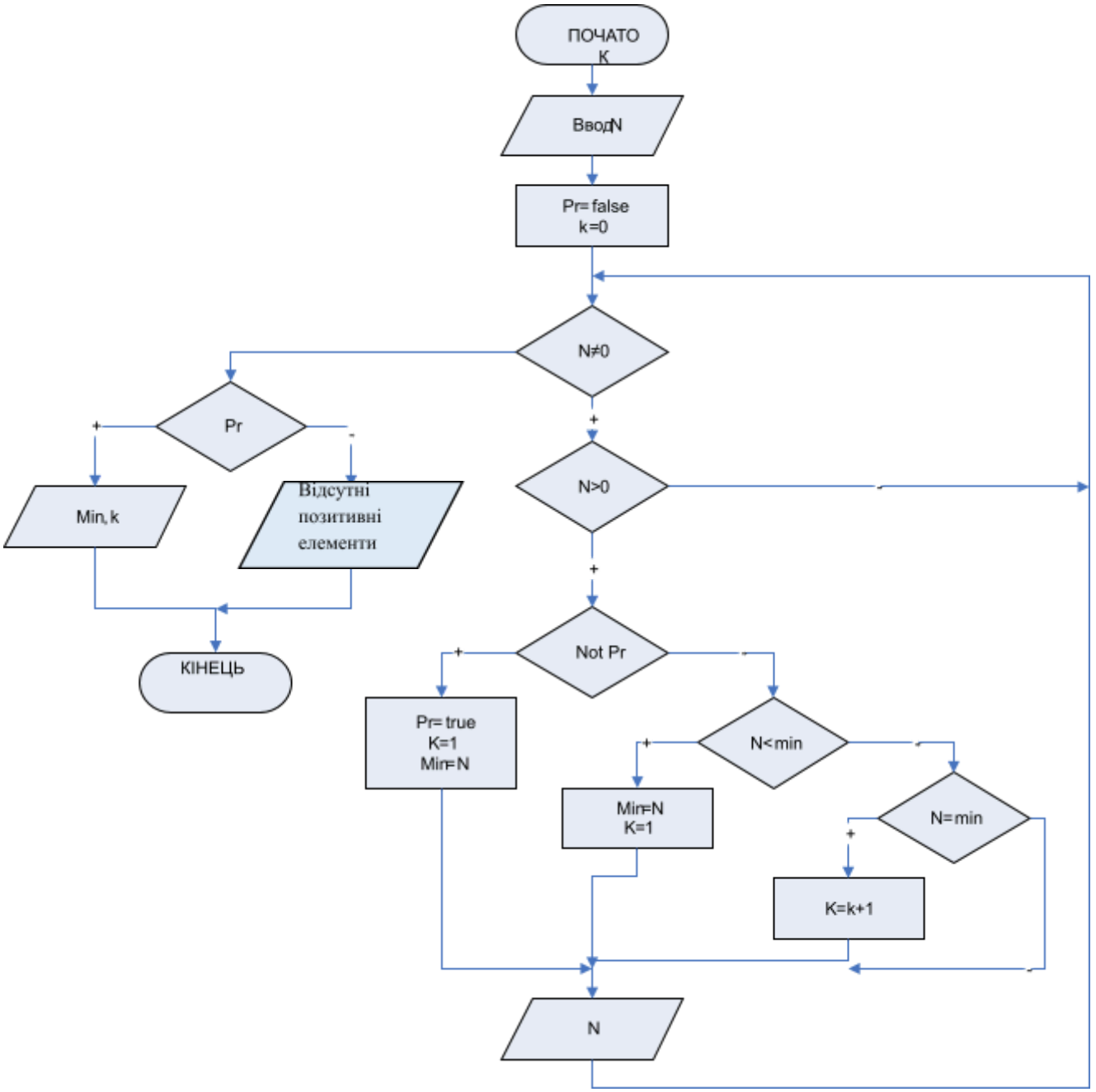


Рис. 7.19. Блок-схема знаходження мінімального додатного числа послідовності

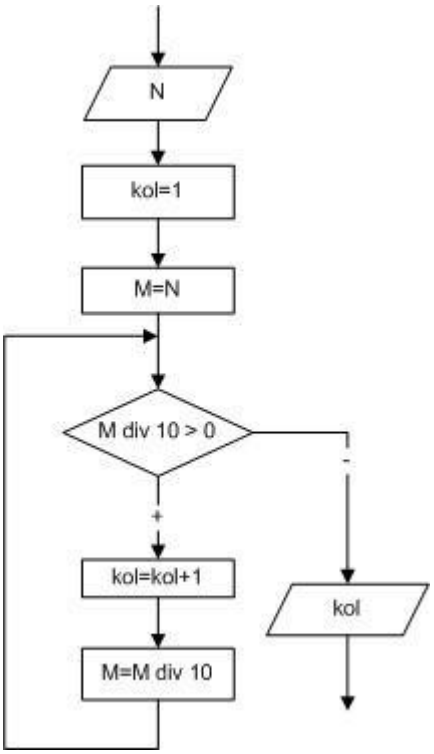


Рис. 7.20. Визначення кількості цифр в числі

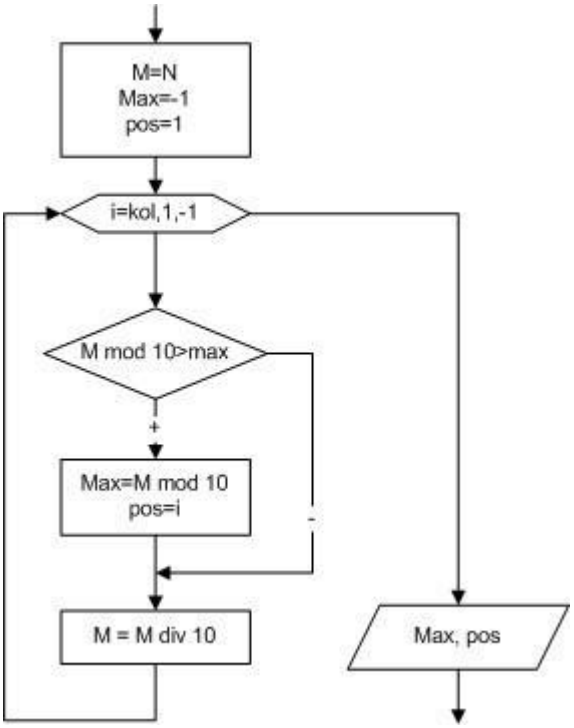


Рис. 7.21. Визначення максимальної цифри в числі і її положення

Якщо цифри числа відомі, визначити найбільшу з них не складе труднощів. Алгоритм знаходження максимального значення в послідовності цифр виглядає наступним чином. У комірку, в якій буде зберігатися максимальний елемент (max), запишіть значення менше, ніж будь-який з елементів послідовності (в нашому випадку max=-1, так як цифри числа знаходяться в діапазоні від 0 до 9). Потім елементи послідовності

порівнюються зі значенням комірки `max`, якщо є елемент, що перевищує значення передбачуваного максимуму, то комірці max потрібно присвоїти значення цього елемента і, відповідно, запам'ятати її номер в послідовності (в нашому випадку змінної `pos` присвоюється значення параметра циклу і).

В алгоритмі пошуку мінімуму спочатку запишіть в змінну `min` значення, яке більше будь-якого елемента послідовності заздалегідь. Потім порівняйте всі елементи послідовності з `min`, якщо ми зустрінемо значення менше `min`, ми записуємо його в змінну `min`.

Порівняння та використання рекурсії та циклу.

Рекурсія і цикл — це дві різні техніки для повторення дій в алгоритмах, кожна з яких має свої особливості. Ось основні відмінності між ними:

1. Структура виконання

- Рекурсія : Функція викликає сама себе для виконання повторюваних завдань. Кожен виклик рекурсії створює новий контекст виконання (нову копію функції), що зберігає поточний стан і параметри.

- Цикл : Повторення дій здійснюється через оператори циклу (наприклад, `for`, `while`), де одна і та ж частина коду виконується багаторазово в межах одного контексту виконання без створення нових функцій.

2. Завершення

- Рекурсія : Має базовий випадок, який визначає, коли рекурсія повинна зупинитися. Якщо базовий випадок не визначений або некоректний, це може призвести до нескінченної рекурсії і помилки через переповнення стека.

- Цикл : Завершення циклу визначається умовою виходу. Якщо умова не виконується, це може спричинити нескінченний цикл , що навантажує процесор.

3. Пам'ять

- Рекурсія : Кожен виклик функції додає новий запис до стека викликів, що може призвести до переповнення стека , якщо глибина рекурсії занадто велика.

- Цикл : Використовує фіксовану кількість пам'яті для збереження змінних циклу, тому більш оптимальний з точки зору пам'яті для великих повторень.

4. Застосування

- Рекурсія : Часто використовується для вирішення задач, які мають природну рекурсивну структуру (дерева, графи, математичні задачі на поділ), наприклад, обхід дерева, пошук шляху, обчислення факторіалу або чисел Фібоначчі.

- Цикл : Зазвичай використовується для простих ітераційних процесів, таких як обхід масивів, повторювані обчислення, операції над послідовностями даних.

5. Читабельність коду

- Рекурсія : Може зробити код більш елегантним і коротким , особливо для задач із природною рекурсивною структурою.

- Цикл : Часто більш простіший для розуміння і не потребує спеціальних умов для зупинки, тому його легше налагоджувати.

Приклад:

- Рекурсія (факторіал числа):

```python
def factorial(n):
if n == 1:
return 1
else:
return n * factorial(n - 1)
```

- Цикл (факторіал числа):

```python
def factorial(n):
result = 1
for i in range(1, n + 1):
result *= i
return result
```

Таким чином, рекурсія краще підходить для задач з вкладеними структурами, тоді як цикл більш ефективний у випадках з великою кількістю ітерацій. Обидві техніки мають свої плюси та мінуси і вибір залежить від конкретної задачі.

Контрольні питання

1. Який цикл має назву цикл з передумовою?
2. Який цикл має назву цикл з післяумовою?
3. Яке використання мають цикли з передумовою?
4. Як використовуються цикли з післяумовою?
5. Які вимоги до тіла циклу та початкових значень?
6. Коли можна використовувати безумовний цикл?
7. Як працює безумовний цикл?
8. Яким чином замінити безумовний цикл на умовний?
9. Наведіть приклад циклу з відомим числом повторень
10. Опишіть алгоритм знаходження мінімального (максимального) значення змінної.

Лекція 9. Розробка алгоритмічних структур засобами електронних таблиць.

Практично будь -які цикли та рекурсії можна виконати за допомогою функцій IF та її різновидів. Розглянемо функцію IF та інші.

9.1 Функція IF для розробки алгоритмів

IF (ЯКЩО)

IF(Logcal_test; Value_if_true; Value_if_false) – перевіряє, чи виконується умова, і повертає одне значення, якщо вона виконується, і інше значення, якщо не виконується.

Ім'я аргументу

Опис аргументів функції

Logcal_test (обов'язковий) це будь-яке значення або вираз, який при обчисленні дає значення **TRUE** або **FALSE**.

Value_if_true (не обов'язковий) результат, якщо умова виявиться істинною.

Value_if_false (не обов'язковий) результат, якщо умова виявиться хибною.

Зауваження:

- якщо потрібно використати текст у формулах, візьміть його в лапки (наприклад, **"Текст"**). Виняток становлять лише значення TRUE та FALSE, які Excel інтерпретує автоматично.
- якщо не вказувати результати, які повертатиме формула, то вона буде повертати TRUE або FALSE в залежності від того яке значення при обчисленні дасть логічний вираз.

Приклади використання функції IF.

Приклад №1: перевірити, чи має студент допуск до іспиту? Щоб отримати допуск, студенти групи повинні успішно здати залік.

Розв'язок:

- починаємо складати формулу;
- в якості першого аргументу функції IF виступає наступна умова – **C3 = "залік"**;
- далі вказуємо результати які повертатиме функція, якщо умова **TRUE – "допущений"**, якщо умова **FALSE – "не допущений"**;
- для отримання результату натискаємо клавішу **Enter**.

=IF(C3 = "залік";"допущений";"не допущений")

Приклад №2: встановити нову ціну зі знижкою 30%, якщо товар на полиці магазину зберігається протягом п’яти діб.

Розв’язок:

- починаємо складати формулу;
- в якості першого аргументу функції IF виступає наступна умова – **C3>=5**;
- далі вказуємо результати які повертатиме функція, якщо умова **TRUE** – **D3-D3*30%**, якщо умова **FALSE** – **D3**;
- для отримання результату натискаємо клавішу **Enter**.

=IF(C3>=5;D3-D3*30%;D3)

ФУНКЦІЯ IF з декількома умовами.

Задачі з однією логічною умовою – це окремий випадок, зазвичай при розв’язанні задач потрібно враховувати декілька варіантів рішення. Наприклад, треба визначити яке число ми отримали: позитивне, негативне або нуль.

В цьому випадку потрібно використовувати вкладені один в одного оператори IF. Це буде виглядати наступним чином:

=IF(Local_test; Value_if_true; IF(Local_test; Value_if_true; Value_if_false))

Тобто, якщо умова дає значення **TRUE** то оператор **IF** повертає значення другого аргументу, якщо **FALSE** оператор перевіряє наступну умову і так далі за ступенями вкладання.

Взагалі, в залежності від того як буде задана умова в операторі **IF** можна скорегувати де будуть знаходитися вкладені оператори у формулі.

Приклад: визначити яке число ми отримали: позитивне, негативне або нуль.

Розв’язок:

- починаємо складати формулу;
- в якості першого аргументу функції **IF** виступатиме наступна умова – **C3=0**;
- значення другого аргументу, коли умова дає **TRUE** буде – **0**;
- в якості значення третього аргументу, коли умова дає **FALSE** виступатиме вкладена функція **IF** зі своїми аргументами:
  - перший аргумент – умова – **C3>0**;
  - значення другого аргументу – **“позитивне”**;
  - значення третього аргументу – **“негативне”**.

○

=IF(C3=0; 0; IF(C3>0; “позитивне”; “негативне”))

Розширення функціоналу IF за допомогою операторів AND та OR.

Функція **IF** дає змогу порівняти значення з очікуваним результатом, використовуючи логічну операцію. Для цього вона перевіряє умову та повертає результат, якщо перевірка дала значення **TRUE** або **FALSE**.

=IF(якщо умова – істина, то виконати наступну дію, інакше – іншу дію)

Але, бувають випадки, коли треба перевірити одночасно декілька умов, і щоб кожна з них була істиною, або хоча б одна з них була істиною, або взагалі перевірити чи не відповідають умови певним умовам. В цих випадках доцільно використовувати разом з функцією **IF** допоміжні оператори **AND (І)** чи **OR (АБО)**.

AND (І)

AND(Logical1; Logical2;) – повертає значення TRUE, якщо всі її аргументи мають значення TRUE, і повертає значення FALSE, якщо хоча б один аргумент має значення FALSE.

Ім’я аргументу

Опис аргументів функції

Logical1 (обов’язковий) перша умова, яку потрібно перевірити, дає результат **TRUE** або **FALSE**.

Logical2 (не обов’язковий) додаткові умови (не більше 255), які потрібно перевірити та які можуть давати результат **TRUE** або **FALSE**.

Таблиця істинності функції AND

Logical1	Logical2	Result
True	True	True
True	False	False
False	True	False
False	False	False

OR (АБО)

OR(Logical1; Logical2;) – повертає значення TRUE, якщо принаймні один аргумент має значення TRUE, і повертає значення FALSE, якщо всі аргументи мають значення FALSE.

Ім’я аргументу

Опис аргументів функції

Logical1 (обов'язковий) перша умова, яку потрібно перевірити, дає результат **TRUE** або**FALSE**.

Logical2 (не обов'язковий) додаткові умови (не більше 255), які потрібно перевірити та які можуть давати результат **TRUE** або **FALSE**.

Таблиця істинності функції OR

Logical1	Logical2	Result
True	True	True
True	False	True
False	True	True
False	False	False

Приклад №1: Перевірити, чи пройдено тестування? Для цього, на кожному з іспитів студент повинен набрати не менше 70 балів.

Розв'язок:

- починаємо складати формулу;
- так як нам треба перевірити одразу декілька умов і кожна з них повинна давати істину, скористаємося оператором **OR** і умова в нашому рівнянні буде записана наступним чином – **AND(C3>=70; D3>=70; E3>=70)**;
- далі вказуємо результати які повертатиме функція, якщо умова **TRUE – "пройдено"**, якщо умова **FALSE – "не пройдено"**;
- для отримання результату натискаємо клавішу **Enter**.

=IF(AND(C3>=70; D3>=70; E3>=70); "пройдено"; "не пройдено")

Приклад №2: Перевірити, чи пройдено тестування? Для цього, хоча б на одному з іспитів студент повинен набрати не менше 70 балів.

Розв'язок:

- починаємо складати формулу;
- знов ж таки треба перевірити декілька умов, але тепер хоча б одна з умов повинна давати істину, отже скористаємося оператором **OR** і умова в нашому рівнянні буде записана наступним чином – **OR(C3>=70; D3>=70; E3>=70)**;
- далі вказуємо результати які повертатиме функція, якщо умова **TRUE – "пройдено"**, якщо умова **FALSE – "не пройдено"**;
- для отримання результату натискаємо клавішу **Enter**.

=IF(OR(C3>=70; D3>=70; E3>=70); "пройдено"; "не пройдено")

Функція IFS (ЯКЩОС) працює аналогічно до IF, але дозволяє *перевіряти більше однієї умови.*

Наприклад,

· якщо студент набрав менше 60 балів, він отримує оцінку «Незадовільно»,

· якщо він набрав від 60 до 75 балів — «Задовільно»,

· якщо він набрав від 75 до 90 балів —«Добре»,

· якщо він набрав більше 90 балів — «Відмінно».

Формула виглядатиме так:

=IFS(A2<60,«Незадовільно»,A2>=60,A2<75,»Задовільно»,A2>=75,A2<90,«Добре»,A2>=90,«Відмінно»),

де **A2** — це клітина з кількістю балів студента.

**Функція IFERROR (ЯКЩОПОМИЛКА)* дозволяє обробити помилки, які можуть виникнути в процесі роботи з формулами. IFERROR повертає вказане значення, якщо обчислення формули — помилки; в іншому разі вона повертає результат формули.

*Функція IFS дає можливість перевірити до 127 умов. Але вкладати багато умов не варто: такі вирази важко створювати та перевіряти.

Обчислюються такі типи помилок: **#N/A, #VALUE!, #REF!, #DIV/0!, #NUM!, #NAME?,** або **#NULL!.**

Синтаксис функції IFERROR:

IFERROR(значення;значення_якщо_помилка)

значення — обов’язковий аргумент, який перевіряється на наявність помилок.

значення_якщо_помилка — обов'язковий аргумент. Значення, яке повертається, якщо обчислення формули — помилка.

Наприклад, якщо ми хочемо поділити значення в одній клітині на значення в іншій, але значення у другій клітині дорівнює нулю, то з'явиться помилка #DIV/0!.

=IFERROR(A1/B1,0)

Використовуйте функції **IF, IFS та IFERROR**, щоб спростити обробку даних і автоматизувати багато процесів в електронних таблицях Excel.

9.2 Формули в Microsoft Excel

Усі обчислення в електронних таблицях ведуться автоматично за допомогою формул, тобто зміна будь-якого вхідного даного одразу ж веде до переобчислення усієї таблиці.

Основне правило використання формул полягає у тому, що якщо значення комірки залежить від значень інших комірок табл иці, то завжди

необхідно використовувати формулу, навіть коли операцію легко зробити «усно». Це гарантує, що подальше редагування таблиці не порушить її цілісності та правильності обчислень, які в ній проводяться.

Будь-яка формула обов'язково починається зі знаку «=». Якщо про цей знак забути, то введені у комірці дані сприйматимуться програмою як звичайний текст. Зрозуміло, що така формула працювати не буде.

Дуже важливо завжди пам'ятати таку властивість електронної таблиці:

–якщо вмістом відміченої комірки є значення, то, і в самій комірці, і в полі введення рядка формул воно відображується однаково, тобто у вигляді значення (рис. 9.1);

–

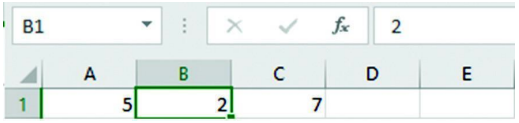


Рис. 9.1. Відображення значення у комірці та рядку формул

– якщо вмістом відміченої комірки є формула, то в самій комір-ці відображується результат обчислення, тобто значення формули, а в рядку формул відображується сама формула (рис. 9.2).

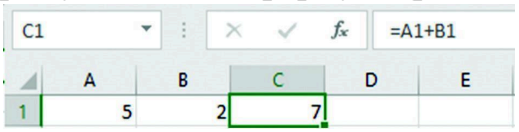


Рис. 9.2. Відображення формули у комірці та рядку формул

Формула може містити числові константи, посилання на комірки і функції Microsoft Excel, які з'єднуються знаками математичних операцій. Отож комірка, яка вміщує формулу, є залежною від числа, що знаходиться в іншій комірці. Значення, яке відображається у комірці з формулою, перераховується під час зміни значення комірки, на яку робиться посилання у формулі.

Посилання на комірку можна задати різними способами. По-перше, адресу комірки можна задати вручну. Інший спосіб полягає у виділ енні необхідної комірки або діапазону, адресу якого необхідно задати. Комірка або діапазон водночас виділяються пунктирною рамкою.

Усі діалогові вікна програми Microsoft Excel, що містять поля введення адрес або діапазонів комірок, містять кнопки, натиснення яких призводить до згортання діалогового вікна до мінімально можливого розміру, що полегшує вибір необхідної комірки або діапазону (рис. 9.3).

Рис. 9.3. Згорнуте діалогове вікно

Для редагування формули необхідно встановити курсор мишки на комірці з формулою і двічі натиснути ліву клавішу мишки або активізувати

відповідну комірку та натиснути клавішу F2. Водночас комірки, від яких залежить значення формули, виділяються на робочому аркуші кольоровими рамками, а самі посилання відображаються у комірці та в стрічці формул тим же кольором. Це полегшує редагування та перевірку правильності формул.

Щоб побачити усі формули у комірках таблиці (рис. 9.4), потрібно задати режим відображення формул у комірках послідовністю команд: Файл – Параметри – Додатково – група Параметри відображення цього аркуша – Формули в клітинках замість обчислених результатів

	A	B	C	D
1	5	2	6	=СУММ(A1:C1)
2				=A1*B1
3				

Рис. 9.4 Таблиця Microsoft Excel у режимі відображення формул

Щоб побачити результати обчислень, потрібно вимкнути режим відображення формул.

Формули в Microsoft Excel поділяють на чотири групи: арифметичні, порівняльні, текстові та адресні. Формули кожної групи мають свій набір операторів і застосовуються по-різному.

Арифметичні формули – під час виконання обчислень оперують числами, адресами комірок, функціями, які об'єднуються арифметичними операторами (+, -, *, /, %, ^). Дужки дають змогу змінювати стандартний порядок виконання дій. Результатом виконання арифметичних операцій завжди є число. Наприклад, =(A1+B3)*C2^2.

Порядок виконання обчислень, що задаються операторами, в електронних таблицях відповідає порядку, визначеному правилами математики:

- **1** (найвищий) – унарний мінус (-);
- **2** – піднесення до степеня (^);
- **3** – множення та ділення (*, /);
- **4** – додавання та віднімання (+, -)

Формули порівняння – використовуються у функціях, які порівнюють два чи більше чисел за допомогою операторів порівняння (=, <, >, <=, >=, <> – не дорівнює). Якщо значення порівняння – істина, то результату формули присвоюється логічне значення «істина», що еквівалентно до будь-якого ненульового значення, в іншому разі формула повертає логічний результат «хибність», еквівалентний до значення нуль.

Текстові формули – це формули, призначення яких полягає в об'єднанні послідовності символів з декількох комірок в одну. У текстових формулах використовується оператор & (*амперсанд*), який оперує текстовими комірками, текстовими стрічками у лапках і результатами текстових функцій. Наприклад, комірка A1 містить текст «Іванов», а комірка B1 – «Іван», тоді під

час використання даного оператора у комірці С1 можна виконати операцію об'єднання =A1&B1. Результатом виконання даної операції буде словосполучення «Іванов Іван».

Адресні формули – формули, що формують з двох посилань на комірки чи діапазони комірок одне об'єднане посилання, використовуючи адресні оператори:

- : (двокрапка) – оператор, що визначає діапазон комірок між верх-ньою лівою і нижньою правою. Наприклад, A6:P14;
- , (кома) – оператор об'єднання комірок декількох діапазонів. На-приклад, A6:P14,Q15:T20, W25 – результатом будуть усі вказані комірки;
- (пробіл) – оператор перехрещення, який вказує на спільні комірки діапазонів, що перехрещується. Наприклад, A1:C5 B4: D5.

У цьому разі діапазон комірок B4:C5 буде спільним.

Основні правила для побудови формул такі:

- формула записується лише в рядок;
- аргументи функцій обов'язково беруться у дужки;
- дужки у формулах можуть бути тільки круглими;
- аргументи функції відокремлюються один від іншого комою (якщо було виконане відповідне налаштування);
- знак множення у формулах пропускати не можна;
- якщо у формулі використовується текст, то він має бути взятий у подвійні лапки.
-

9.3 Пошук помилок у формулах

Якщо під час введення формули допущені помилки, результатом її обчислення буде повідомлення, яке з'явиться у комірці (табл. 9.1).

Таблиця 9.1 Типові помилки, які трапляються під час введення та роботи з даними та формулами

Помилка	Причини виникнення
#####	1. Введене числове значення або результат обчислення формули не вміщується в комірці. 2. Під час визначення числа днів між двома датами, або кількості годин між двома часовими проміжками виходить негативне значення.
#VALUE (#ЗНАЧЕННЯ)	1. Замість числового або логічного значення введений текст, і Microsoft Excel не може перетворити його до потрібного типу даних. Наприклад, якщо в комірці A1 міститься число 8, в комірці B1 – текст «Василь», а в комірці C1 – формула = A1 + B1, то в C1 буде виведена помилка. 2. Використана неправильна розмірність матриці даних у відповідній функції.

#DIV/0! (#ДІЛЕННЯ/0!)	Ділення на 0. Таке повідомлення виникає у тих випадках, коли значення дільника операції ділення виявляється рівним нулю.
#NAME? (#ІМ'Я?)	1. Програма не може розпізнати текст, уведений у формулу. Наприклад, неправильне ім'я функції =КОРІНЬ(tg(1)+1). Правильне рішення: =КОРІНЬ(TAN(1)+1). 2. У посиланні на діапазон комірок пропущений знак двокрапки (:). 3. У формулу введений текст, який не укладений у подвійні лапки.
# N / A (# Н / Д)	Дані комірки одного з аргументів у цей момент недоступні.
#REF! (#ПОСИЛАНЯ!)	Неправильне посилання на комірку
#NUM! (#ЧИСЛО!)	1. Помилка в аргументі функції. Наприклад, аргумент функції у формулі =КОРІНЬ(-5) – від'ємний. 2. Неможливо обчислити результат формули, або він надто великий чи малий для коректного відображення у комірці
Виявлені помилки у введеній формулі	Таке повідомлення бачимо, наприклад, під час уведення помилкової формули =10+*2, в якій указано дві операції підряд. При цьому пропонується варіант виправлення формули. Якщо користувач погоджується із запропонованим, то натискає кнопку **Так**. Якщо користувач натискує кнопку **Ні**, то це означає, що виправлення він збираєтесь внести самостійно.
Невідповідність дужок	Таке повідомлення виникає у тих випадках, коли у введеній формулі порушено баланс між дужками, які відкриваються і закриваються. Наприклад: =(2+3*4.

Рис. 9.5. Отримання інформації про помилку у таблиці

Для отримання інформації про помилку необхідно викликати меню кнопки зі знаком оклику, що з'явилась зліва від комірки із помилкою (рис. 9.5). За допомогою цього меню можна отримати детальну інформацію про помилку, виконавши команду **Довідка з цієї помилки**, а також відстежити всі етапи розрахунків та встановити параметри перевірки помилок.

Під час пошуку помилок у таблицях із великою кількістю формул використовується *функція відслідковування залежностей*, яка дає змогу у графічному вигляді представити на екрані зв'язки між різними комірками робочої книги. Комірка є *залежною*, якщо вона містить формулу з посиланням на активну комірку. *Впливаючою* називається комірка, на яку посилається формула в активній комірці.

Команди для встановлення залежностей і пошуку помилок розташовані на вкладці стрічки **Формули** у групі **Аудит формули** (рис. 9.6).

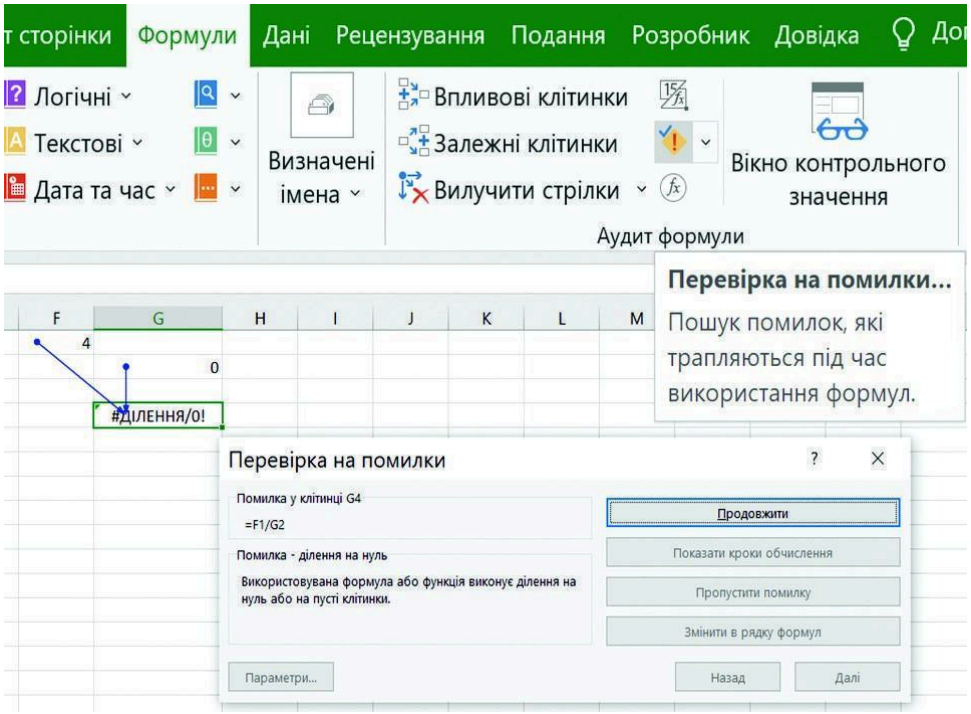


Рис. 9.6. Пошук помилок та встановлення залежностей

Якщо, наприклад, виділити комірку із формулою і виконати команду **Впливові клітинки**, то з'являться лінії трасування зі стрілками, які з'єднують залежні комірки у таблиці.

Для пошуку і аналізу усіх помилок у книзі використовується команда **Перевірка на помилки**…, яка відкриває однойменне діалогове вікно.

За звичайних умов під час введення у комірку формули електронна таблиця налаштована так, щоб сама формула вводилась і відображалась лише у рядку формул, а у комірці ж виникав результат обчислення. Під час пошуку помилок ці параметри можна змінити послідовністю команд **Файл** ⇨ **Параметри**… **Додатково** ⇨ група **Параметри відображення цього аркуша** **Відображати формули в клітинках замість обчислених результатів**. В

результаті у всіх комірках аркуша відобразяться формули, що значно полегшить перевірку.

9.4 Копіювання формул

Адресація в Microsoft Excel. Копіювання формул і автоматичне переобчислення. В процесорах електронних таблиць у формулах та функціях для посилання на комірки і блоки комірок використовують імена комірок, як адреси. Існує три види адрес: *відносні, абсолютні* та *змішані*.

Відносна адресація. За замовчуванням посилання на комірки розглядаються як відносні. Вони і мають вигляд звичайної адреси (наприклад, A1 або A1:C8) і змінюються під час копіювання формул відповідно до відносних розташувань вихідної комірки і створюваної копії.

Наприклад, комірка C1 містить формулу = A1+B1. Необхідно скопіювати цю формулу у дві комірки, які знаходяться нижче C1. Це можна зробити і за допомогою буфера обміну (Ctrl+C, Ctrl+V), і шляхом *автозаповнення*.

Під час автозаповнення необхідно:

- виділити комірку C3;
- сумістити з маркером заповнення (квадрат у правому нижньому кутку комірки) вказівник мишки (набуде форми маленького чорного хрестика);
- натиснути ліву клавішу мишки і, утримуючи її, протягнути вказівник на дві комірки вниз.

Тепер візьмемо до уваги вигляд формул у комірках C2 і C3. Як не дивно, але вони усі різні і мають вигляд =A2+B2 і =A3+B3 відповідно. Це і означає, що під час копіювання автоматично відбулося налаштування формул на нове місце розташування *відносно* вихідних комірок. Принцип налаштування, у цьому разі, полягає в тому, що, як і основна формула, нові формули набувають значення аргументів на тій же відстані зліва від них (рис. 9.7).

Рис. 9.7. Використання відносної адресації

Аналогічно, якщо у комірці B2 є посилання на комірку A3, то можна сказати, що посилання здійснюється на комірку, яка розташована на один стовпчик лівіше і на одну стрічку нижче цієї. Якщо формула буде копіюватися в іншу комірку, то таке відносне посилання збережеться. Наприклад, під час копіювання формули у комірку H27 посилання буде продовжувати показувати на комірку, яка розташована на один

стовпчик лівіше і на одну стрічку нижче, у цьому разі на комірку G28.

Під час використання стандартних способів копіювання вмісту комірки за допомогою буферу обміну, табличний процесор також автоматично змінить посилання на комірки.

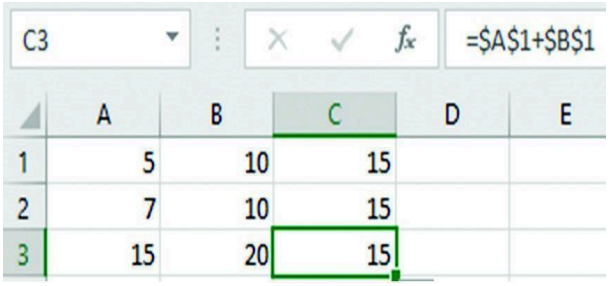


Рис. 9.8. Використання абсолютної адресації

Абсолютна адреса. Якщо посилання на комірку не повинно змінюватися під час копіювання, то використовують абсолютну адресу комірки. Для вказання абсолютної адреси застосовується символ «$», наприклад, A1, тобто додається символ «$» перед позначенням стовпця і номером рядка (рис. 9.8). Зміну вигляду посилання виконують також за допомогою клавіші **F4** при виділенні необхідної адреси у рядку формул.

Змішані адреси. У адресі комірки абсолютними можуть бути номери рядка і стовпчика як окремо, так і разом. Наприклад, відносні та абсолютні адреси комірки A1 мають вигляд A1 і A1 відповідно. А в адресах A$1 і $A1 абсолютним є лише один з компонентів.

Нестандартні імена комірок. Імена комірок використов ують як позначення змінних у складі формул. Імена комірок можуть бути стандартними і нестандартними. Стандартні імена, як було зазначено, становлять адреси комірок, побудовані шляхом поєднання позначень стовпчика та рядка. Для будь-якої комірки можна обрати також довільне нестандартне ім’я.

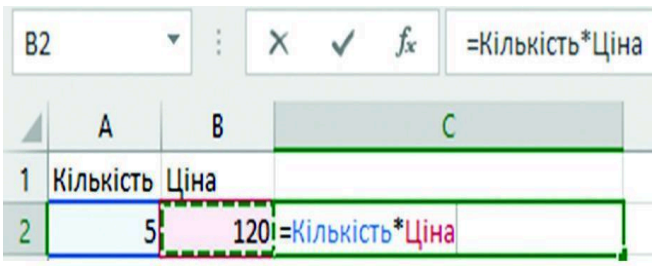


Рис. 9.9. Формула із нестандартними іменами

Зручність нестандартних імен пов'язана з їх звичністю, а також з можливістю надання їм змістовного звучання. Наприклад: X, Y, Результат, Підсумок тощо. А це значно полегшує побудову та сприйняття формул (рис. 9.9).

Для створення нестандартного імені виділяють потрібну комірку, а потім виконують команду контекстного меню **Визначити ім'я**… Відкриється діалогове вікно **Нове ім'я** (рис. 9.10), у якому вказується нове ім'я.

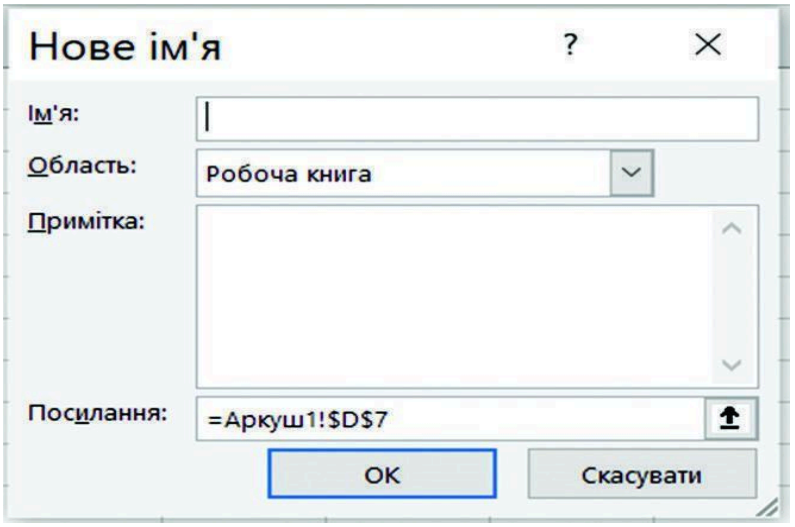


Рис. 9.10. Діалогове вікно Нове ім'я

На *нестандартні імена* накладаються такі обмеження:

- вони не можуть містити пропусків (пробілів);
- вони не мають збігатися з адресою будь-якої іншої комірки;
- вони можуть бути лише абсолютними;
- великі та малі літери в іменах не розрізняються у будь-якому алфавіті.

Нестандартні імена мають такі властивості:

- якщо виділити комірку із нестандартним іменем, то саме воно буде тепер відображатись у полі **Ім'я** рядка формул;
- поле **Ім'я** рядка формул фактично становить список вже наявних нестандартних імен (рис. 9.11). Оскільки нестандартні імена на робочому аркуші себе ніяк не проявляють, то цей список може використовуватись для виділення комірок з нестандартними іменами;

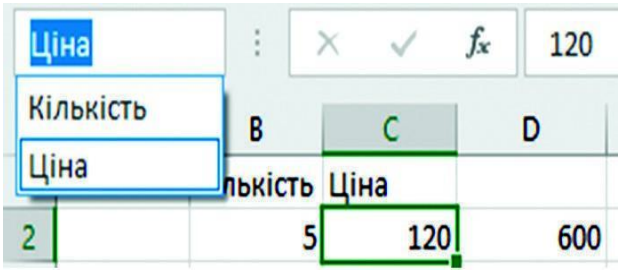


Рис. 9.11. Список нестандартних імен

– нестандартні імена можуть бути присвоєні не лише коміркам, а й діапазонам;

– надані нестандартні імена є глобальними, тобто вони дійсні на будь-яких аркушах книги, але їх можна створити і локальними, тобто дійсними у межах лише одного аркуша. Для цього до складу імені потрібно включати також і ім'я аркуша. Наприклад: Лист3!Ціна. Водночас нестандартні імена аркушів беруться в одинарні лапки. Наприклад: 'Результати'!Ціна;

– одній і тій же комірці або діапазону можна надати довільну кількість імен.

Вилучення або зміна наявного нестандартного імені здійснюється у діалоговому вікні **Диспетчер імен** (рис. 9.12), яке відкривається

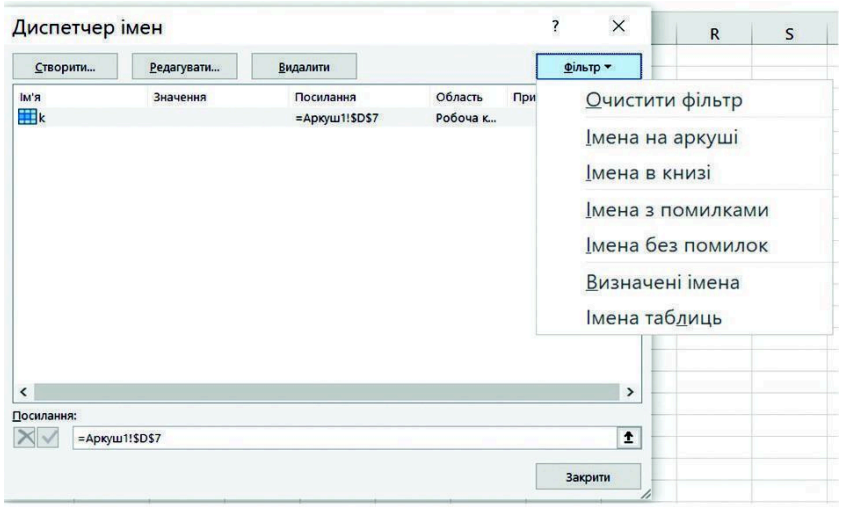


Рис. 9.12. Вікно Диспетчер імен

послідовністю команд: вкладка **Формули** ⇨ кнопка **Визначені імена** ⇨ кнопка **Диспетчер імен**. Для виклику діалогового вікна **Диспетчер імен** також можна використати комбінацію клавіш **Ctrl+F3**.

Контрольні питання

1. З яких елементів складається графічний інтерфейс програми Microsoft Excel?
2. Як створити випадаючий список?

3. Які існують операції редагування даних в електронних таблицях?
4. Яку послідовність команд необхідно виконати, щоб побачити всі формули у комірках?
5. Які основні правила для побудови формул?
6. Для чого використовується функція відслідковування залежностей?
7. Що таке відносна адресація?
8. Що таке абсолютна адресація?
9. Для чого використовуються і як задаються нестандартні імена комірок?
10. Якими способами можна ввести функцію?
11. Які категорії функцій існують?

Лекція 10. Підготовка та аналіз даних за допомогою інструментів форматування.

10.1 Швидке форматування

Якщо одна із комірок має потрібний формат і таке ж оформлення має бути і у інших комірок, то для того, щоб застосувати цей формат, необхідно:

1. Виділити комірку з потрібним форматуванням.
2. Натиснути кнопку **Формат за зразком** на вкладці **Основне**.
3. Виділити діапазон комірок, щоб застосувати форматування.

Для того, щоб застосувати формат у декількох місцях, потрібно кнопку **Формат за зразком** натиснути двічі.

У Microsoft Excel також пропонуються готові стилі окремих комірок та всієї таблиці для швидкого багаторазового форматування. Розташовані вони на вкладці **Основне** у групі **Стилі** в галереях **Формат таблиці** (рис. 10.1) та **Стилі комірок** (рис. 10.2).

Рис. 10.1. Галерея стилів таблиці

За необхідності можна створити власний стиль командами **Створити стиль таблиці**… та **Створити стиль клітинки**…, що відкривають

відповідні діалогові вікна (рис. 10.2), в яких задаються параметри форматування.

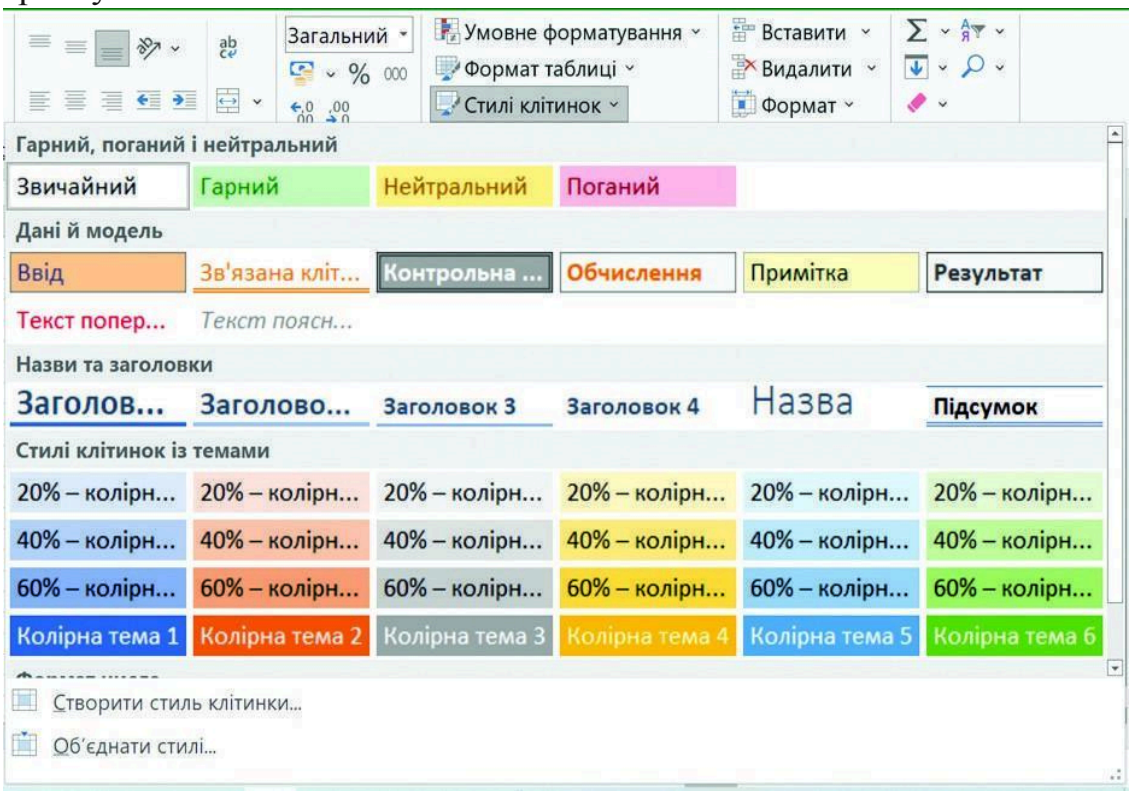


Рис. 10.2. Галерея стилів комірки

Рис. 10.3. Діалогові вікна для створення стилю таблиці та клітинки

10.2 Умовне форматування

Внаслідок застосування формул і табличних даних, які змінюються, результати обрахунків теж стають змінними. Водночас вони можуть значно змінюватись. Щоб покращити сприйняття табличних даних, що змінюються під час створення табличних документів, і полегшити пошук потрібної інформації, може використовуватись специфічне виділення певних характерних табличних значень (мінімальних, максимальних, таких, що перевищують або не досягають певної межі тощо). Специфічне виділення комірки може бути шрифтовим, кольоровим, а також за допомогою границь. Зрозуміло, що за указаних умов таке виділення комірок має бути залежним від величини значень, які в цих комірках зберігаються. Потрібний результат досягається, так званим, умовним форматуванням.

Умовний формат – це такий формат, який Microsoft Excel автоматично застосовує до комірки, якщо виконується вказана умова.

Для встановлення умовного форматування заданої комірки виконують послідовність команд: вкладка **Основне** ⇨ група **Стилі** ⇨ **Умовне форматування**. У меню, що відкриється (рис. 10.4) необхідно вибрати один із встановлених форматів поведінки.

Найчастіше використовуються **Правила виділення клітинок** (див. рис. 10.4) (призначені для виділення значень, що належать деякому діапазону) та **Гістограми** (допомагають розглянути значення

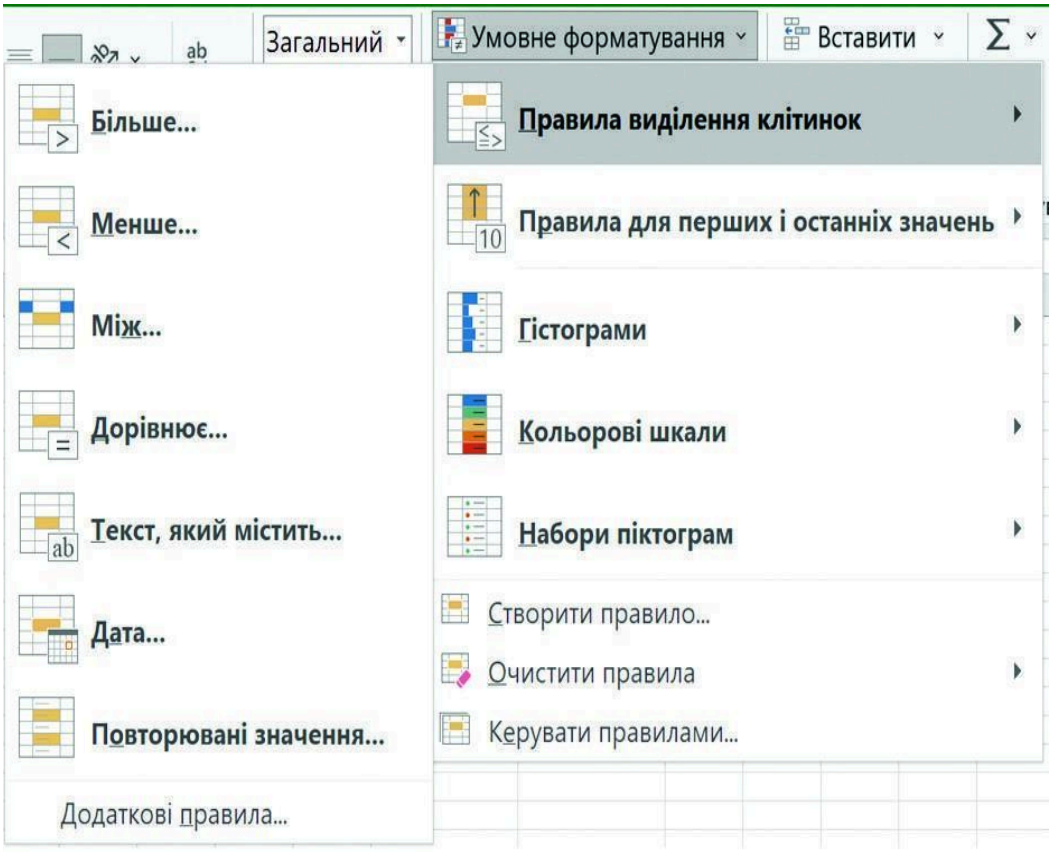


Рис. 10.4. Встановлення параметрів умовного форматування

у комірці щодо інших комірок. Довжина гістограми відповідає значенню у комірці. Що вона довша, то більше значення).

Наприклад, застосуємо різне умовне форматування для однакових значень (рис. 10.5):

Між 4 і 8	Вище середнього	Гістограма	Кольорова шкала	Набори значків
3	3	3	3	3
5	5	5	5	5
9	9	9	9	9
4	4	4	4	4

Рис. 10.5. Застосування умовного форматування для однакових значень

Можна використовувати одночасно декілька правил форматування комірок (до трьох). Застосувати спочатку одне правило, а потім інше. Якщо вони не суперечать одне одному, то формат комірок буде «багатошаровим».

Якщо через певні причини користувача не влаштовують правила вбудовані в Microsoft Excel, то можна створити власне правило. Для цього виконують команду **Створити правило**… (рис. 10.6) та обирають тип правила. Доступні такі типи:

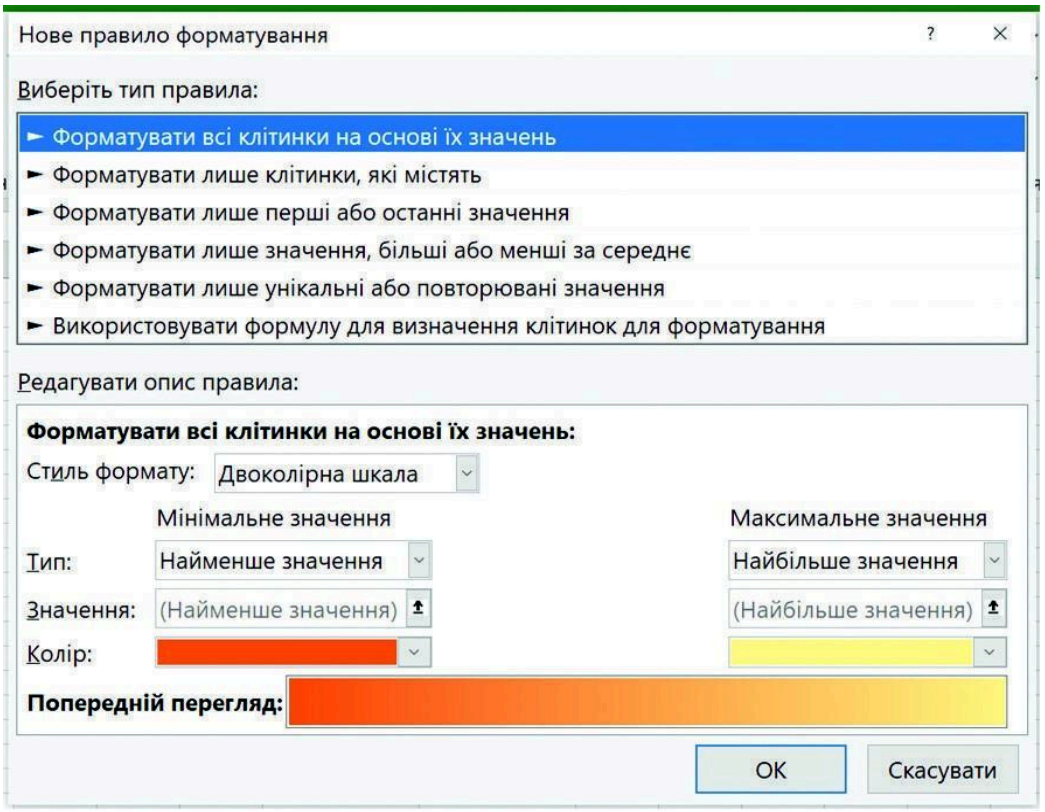


Рис. 10.6. Діалогове вікно для створення нових правил форматування

– **Форматувати всі клітинки на основі їх значень** – означає, що форматування відбуватиметься у вказаному діапазоні на осн ові вказаних

правил діапазону відбору. Наприклад, вказавши мінімальне та максимальне значення, можна встановити колір заливки (двотрьокольоровий градієнт), піктограму або гістограму, що відображатиметься у кожній комірці виділеного діапазону.

– **Форматувати лише клітинки, які містять** – виконуватиме форматування лише тих комірок, які задовольнятимуть вказаним правилам.

– **Форматувати лише перші або останні значення** – форматуватиме лише крайній діапазон (числовий або відсотковий) від вказаного.

– **Форматувати лише значення, більші або менші за середні** – форматуватиме комірки, які містять значення вище або нижче (можливо встановити додаткове відхилення) від середнього значення виділеного діапазону.

– **Форматувати лише унікальні або повторювані значення** – форматуватиме тільки ті комірки, які міститимуть унікальні дані або дані, які дублюються декілька разів.

– **Використовувати формулу для визначення клітинок для форматування** – використовувати форматування згідно з введеною формулою. Формула має повертати значення TRUE або FALSE.

Під час використання умовного форматування, усі правила, які застосовуються до аркуша, зберігаються та додаються до списку правил, то ж іноді це призводить до виникнення конфліктів, тому варто бути уважним під час додавання нових правил.

Видалити окреме правила умовного форматування можна, використавши команду **Очистити правила** (див. рис. 10.4).

Якщо ж на аркуші застосовано багато різноманітних правил умовного форматування, то зручно використовувати **Диспетчер правил умовного форматування** (рис. 10.7), що відкривається такою

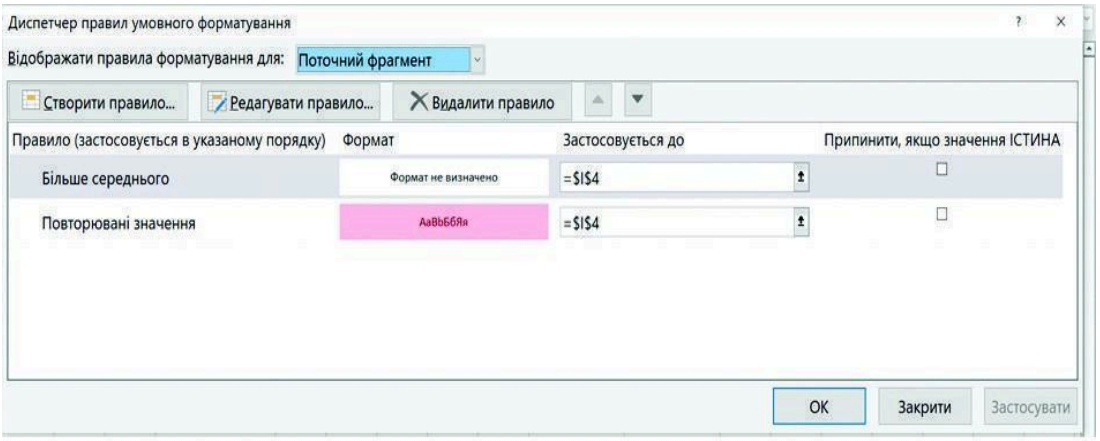


Рис. 10.7. Диспетчер правил умовного форматування

10.3. Інструменти аналізу даних.

Організація графічної інформації

Ділова графіка – це спеціальний клас графічних зображень, що дають змогу представляти у наочній формі числові дані та оздоблювати своєрідними графічними коментарями зображення різного призначення. Найчастіше ділова графіка використовується під час підготовки різноманітних звітів, доповідей, презентацій, статистичних зведень тощо. Доволі часто об'єкти ділової графіки використовуються під час підготовки наукової та навчальної літератури.

Створення діаграм. Діаграми – засіб наглядного представлення інформації, заданої у вигляді чисел. Діаграми створюються на основі чисел, що містяться у робочому аркуші. Одна діаграма може використовувати дані з будь-якої кількості аркушів і навіть з будь-якої кількості робочих книг. Діаграма зберігає зв'язок з даними, на основі яких вона побудована, і під час зміни цих даних відразу змінюється і вигляд діаграми.

Табличні процесори генерують діаграми автоматично, а користувачу лишається тільки вибрати діапазон комірок, вказати тип діаграми та оформити отриманий графічний об'єкт за власним смаком.

Для того, щоб створити діаграму в Microsoft Excel, необхідно:

1. Увести числові дані на аркуш одним із наступних способів:

– безпосередньо в програмі Microsoft Excel;

– скопіювати через буфер обміну;

– імпортувати з документів, що створені у інших програмах (команда Дані Отримання зовнішніх даних).

2. Виділити діапазон даних, на основі яких буде будуватися діаграма.

Під час виділення у діапазон потрібно включати заголовки рядків і стовпчиків, які будуть відображені як підписи рядів даних діаграми.

3. Вибрати потрібний тип діаграми, який визначається розташуванням даних у комірках таблиці.

Для цього необхідно на вкладці Вставлення у групі Діаграми вибрати тип та підтип діаграми або натиснути кнопку та у далі діалоговому вікні Вставлення Діаграми на вкладці Усі діаграми вибрати потрібний тип та підтип діаграми.

Розміщення діаграм. Під час створення діаграми в Microsoft Excel існує два варіанти їх розміщення:

– вставити діаграму безпосередньо на аркуш як один з його об'єктів – вбудована діаграма (виконується за замовчуванням); – створити діаграму на новому аркуші робочої книги.

Для зміни варіанту розміщення діаграми необхідно активізувати діаграму та на вкладці Конструктор Діаграм у групі Розташування натиснути кнопку Перемістити діаграму. Відкриється діалогове вікно, у якому вибирається потрібний варіант розміщення.

Діаграма, що розміщена на окремому аркуші, займає весь цей аркуш. Тому, якщо планується друкувати лише одну діаграму на сторінці, найкраще використати цей спосіб. Якщо необхідно створити багато діаг рам, тоді

можна будувати кожну з них на окремому аркуші. Крім того, цей спосіб дає змогу доволі легко розшук увати потрібну діаграму, надаючи аркушам діаграм відповідні назви.

Типи діаграм

Гістограма – спосіб графічного представлення табличних дискретних даних та наочного порівняння різних величин.

На гістограмах категорії зазвичай відкладаються по горизонтальній осі, а значення – по вертикальній.

– *Гістограма з групуванням* – порівнює значення за категоріями та виводить їх у вигляді вертикальних прямокутників (паралелепіпедів, циліндрів, конусів, пірамід).

Формат даних. Кожний рядок в таблиці відповідає окремому стовпчику на діаграмі. У перший стовпчик таблиці вводяться категорії усіх рядків. Решта стовпчиків мають містити числові дані. Для кожного стовпчика даних можуть вказуватися назви, які автоматично виводяться на діаграмі як підписи осей

– *Гістограма з накопиченням* – демонструє відношення окремих складників до їх сукупного значення, порівнюючи за категоріями внесок кожної величини у загальну суму, та виводить їх у вигляді вертикальних прямокутників з накопиченням.

Такою гістограмою користуються, якщо є декілька рядів даних і потрібно зробити наголос на загальному підсумку.

Формат даних. Дані цього типу діаграм повинні мати ту ж структуру, що і гістограми з групуванням. Однак кожна категорія відповідає лише одному стовпчику, який містить усі дані для визначення категорії і графічно відображає відношення між частинами і цілим значенням

– *Нормована гістограма з накопиченням* – порівнює за категоріями відсотковий вклад кожної величини у загальну суму та виводить їх у вигляді вертикальних прямокутників (паралелепіпедів, циліндрів, конусів, пірамід) з накопиченням.

До такої гістограми звертаються, якщо є три або більше рядів даних і потрібно зробити наголос на їх внеску до цілого, особливо якщо підсумок однаковий для кожної категорії.

Формат даних. Дані цього типу діаграм повинні мати ту ж структуру, що і гістограми з накопиченням. Однак кожна категорія відповідає двом і більше стовпчикам та підсумкові значення однакові для кожної категорії

Об'ємна гістограма – використовується для порівняння даних і за категоріями, і за рядками. Вона містить три осі, які можна змінювати (горизонтальна, вертикальна та вісь глибини).

Формат даних. Кожен рядок у таблиці відповідає декільком стовпчикам на діаграмі, а кожен стовпчик декільком рядкам. У перший стовпчик таблиці вводяться категорії усіх рядків. У перший рядок таблиці вводяться категорії усіх стовпчиків. Для кожного стовпчика і рядка даних можуть вказуватися назви, які автоматично виводяться на діаграмі як підписи осей

Лінійна діаграма. Лінійна діаграма – це гістограма, яку повернуто на 90 вправо. Використовується у разі, коли мітки осей та значення, що виводяться, мають велику довжину.

Графіки. Графіки – тип діаграм, що застосовуються для відображення неперервних даних в єдиному масштабі. Графіки з маркерами – тип діаграм, що застосовуються для відображення неперервних даних з відміченими окремими значеннями.

Графіки зручно використовувати у разі, коли у нижній частині діаграми потрібно відобразити дати для демонстрації хронології розвитку події, а також під час порівняння змін значень у декількох категоріях даних.

Формат даних. У перший стовпчик вводять назви категорій, що представлені на графіку. Решта стовпчиків мають містити числові дані, які відповідають лініям діаграми. На графіку може відображатися один або декілька таких стовпчиків. Для кожного стовпчика можна вказати назву (здебільшого місяць, рік, квартал, дата), яка автоматично з'являється в якості підписів даних на діаграмі. На графіках дані категорій рівномірно розподіляються по горизонтальній осі, а всі значення рівномірно розподіляються вздовж вертикальної осі.

Так само як і для гістограм, розрізняють: *графік, графік з накопиченням, нормований графік з накопиченням та об'ємний графік*

Діаграми з областями. Діаграма з областями підкреслює величину змін з часом та використовуються для акцентування на сумарному значенні. Подібна на розмальований різними кольорами графік.

Кругові діаграми. Кругова діаграма – тип діаграм, що дозволяє графічно представляти дані як сегменти кола або відсоткові частини від цілого значення. Вона демонструє розмір елементів одного ряду даних щодо суми усіх елементів.

Формат даних. Для усіх підтипів кругової діаграми дані мають міститися у двох стовпчиках або рядках таблиці. У першому вказується категорія, а у другому – відповідне числове значення

Розрізняють: *кругову, вторинну кругову (для перегляду невеликих секторів головної кругової діаграми) та розрізану кругову діаграму*. Може бути представлена в дво- або тривимірному вигляді.

Точкові діаграми. Точкова діаграма – відображає відношення між числовими значеннями у кількох рядах даних: один набір числових даних уздовж горизонтальної осі (x), а інший – уздовж вертикальної осі (y). Ці значення поєднуються в єдині точки й відображаються в нерівних інтервалах або групах. Точкові діаграми зазвичай застосовуються для відображення та порівняння наукових, статистичних або інженерних даних.

Поверхневі діаграми. Поверхнева діаграма призначена для знаходження оптимальної комбінації даних з двох наборів. На відміну від діаграм інших типів, у них колір використовується для виділення значень, а не різних наборів даних. Області, що належать до однакових діапазонів виділяються однаковими кольорами та візерунками.

Розрізняють: *дротова (без кольору на поверхні), об'ємна (демонструють зміну значень даних за двома вимірами у вигляді поверхонь), контурна (вигляд зверху).*

Пелюсткові діаграми. Пелюсткова діаграма дає змогу порівнювати значення декількох рядів даних. Водночас категорії відображаються на окремих променях двомірного графіку. Точки даних з таблиці відкладаються на відповідних променях і з'єднуються лініями. Усі промені виходять з центру діаграми.

Формат даних. Дані організовуються у вигляді одного або декількох стовпчиків. Перший стовпчик є необов'язковим. У нього вводяться текстові дані для позначення променів. Решта стовпчики мають містити числові дані, які будуть відкладатися уздовж променів діаграми.

Комбіновані діаграми. Комбінована діаграма використовується для суміщення в одній діаграмі різних типів діаграм.

Автоматичне редагування та форматування діаграм. Щоб не редагувати чи не форматувати діаграму у ручному режимі, користуються попередньо визначеними макетами та стилями, які вже містяться у програмі Microsoft Excel. За потреби макет або стиль можна налаштування самостійно, змінюючи окремі елементи діаграми (наприклад, область діаграми чи область побудови).

У разі застосування попередньо визначеного макета, у діаграмі певний набір елементів впорядкується деяким способом. Для кожного типу діаграми є багато різних макетів.

Microsoft Excel пропонує різноманітні корисні колекції макетів (вкладка Конструктор діаграм група Макети діаграм) і стилів (вкладка Конструктор діаграм група Стилі діаграм) (або експрес-макети та експрес-стилі), з яких не лише можна вибрати потрібні варіанти, але й змінити їх, відредагувавши макет або формат окремих елементів діаграми.

Аналіз даних за допомогою діаграм

Для різних типів діаграм доступні додаткові лінії (наприклад, коридор коливань і лінії тренду), смуги (наприклад, смуги підвищення/зниження та планки похибок), позначки даних та інші параметри.

Додавання лінії тренду. Під час побудови діаграм для даних, що залежить від часу, можна будувати лінію тренду, яка буде відображати тенденцію розвитку даних. У деяких випадках за допомогою лінії тренду можна прогнозувати зміни даних. Окремі набори даних можуть мати декілька ліній тренду.

Для встановлення лінії тренду необхідно активізувати діаграму та виконати послідовність команд Конструктор діаграм Додати елемент діаграми Лінія тренду.

Є такі типи лінії тренда: *лінійна, експоненціальна, логарифмічна, поліноміальна, степенева та лінійна фільтрація.* Тип лінії тренду, який слід вибирати, визначається типом наявних даних.

Лінія тренду отримується найточнішою, коли її величина вірогідності апроксимації (наближення) наближується до одиниці. Під час апроксимації

даних за допомогою лінії тренду величина вірогідності апроксимації розраховується Microsoft Excel автоматично. За необхідності отриманий результат можна відобразити на діаграмі. Для цього необхідно викликати контекстне меню лінії тренду, виконати команду Формат лінії тренду та встановити прапорець Розмістити на діаграмі величину апроксимації.

Лінійна лінія тренду – це пряма лінія, яка наближено описує сукупність даних.

Логарифмічна лінія тренду – добре описує величину, яка спочатку швидко зростає або спадає, а далі поступово стабілізується. Наприклад, прогноз росту, зниження правопорушень у певному регіоні.

Поліноміальна лінія тренду – використовується для опису величин, почергово зростаючих та спадаючих. Вона корисна, на приклад, для аналізу великого набору даних про нестабільну величину. Наприклад, залежність витрати палива від швидкості руху.

Степенева апроксимація – корисна для опису монотонно зростаючої або монотонно спадної величини, наприклад відстань, пройдена автомобілем, що прискорюється.

Експоненціальна лінія тренду – використовується у тому разі, якщо швидкість зміни даних постійно зростає. Наприклад, розр ахунок складних відсотків у аналізі даних.

Лінійне фільтрування дає змогу згладити коливання даних і так наочніше показати характер залежності. ЇЇ застосовують, коли точок є багато і вони дуже розкидані. Дані усереднюються попарно чи в інший спосіб і одержані середні значення використовуються як точки лінії тренду.

Лінійний прогноз – дозволяє за допомогою лінії тренду графічно продемонструвати тенденцію зміни даних. Наприклад, якщо в Microsoft Excel є діаграма, що відображає певні дані за перші декілька років, можна додати до цієї діаграми лінію тренду, яка відображатиме загальні зміни цих даних (зростаючу, спадну або рівну) або очікувані зміни в наступних роках.

Контрольні питання:

1. Поясніть, які переваги має Microsoft Excel перед Microsoft Word у створенні та редагуванні таблиць.
2. Опишіть, що таке робоча книга та робочий аркуш у Microsoft Excel і як вони взаємопов’язані.
3. Назвіть основні типи даних, які можуть зберігатися у комірках Excel, та поясніть їх особливості.
4. Перелічіть способи виділення окремої комірки, діапазону комірок, рядків або стовпців у Microsoft Excel.
5. Поясніть, як вводити дані у комірку та як змінюється їх відображення залежно від типу.
6. Опишіть, як створити та редагувати гіперпосилання у Microsoft Excel, які варіанти гіперпосилань існують.
7. Розкажіть, що таке автозаповнення в Excel і як використовуються спеціальні списки та поняття прогресії.

8. Перерахуйте основні типи діаграм у Microsoft Excel та поясніть їх призначення.

9. Визначте, що таке лінія тренду у діаграмах, які її види існують і як обрати найбільш підходящий.

10. Поясніть, як аналізувати дані в Microsoft Excel за допомогою діаграм та які інструменти для цього використовуються.

Лекція 11. Бази даних. Типи, основні зв’язки, засоби створення.

11.1 Основні поняття баз даних

Загальні відомості про інформаційні системи

Характерною рисою сучасного суспільства є стрімке збільшення обсягів інформації, збільшення вимог до її точності та своєчасності. Традиційні способи збереження, пошуку та обробки інформації вже не задовольняють сучасним вимогам. Паперова технологія фактично вичерпала свої можливості по удосконаленню методів роботи з інформацією. Для цього необхідні більш швидкі та ефективні методи.

Принципово нові можливості у поліпшенні інформаційного обслуговування надають сучасні технології, зокрема, автоматизовані інформаційні системи.

Дані в інформаційній системі можуть зберігатися в **неструктурованому** або **структурованому** вигляді.

- **Неструктуровані** дані – це звичайні текстові документи (статті, реферати, книги і т.д.). Системи, в яких зберігаються неструктуровані дані, не завжди можуть дати конкретну відповідь на запит користувача, а можуть видати текст документи або перелік документів, в яких можна знайти відповідь. Наприклад, у відповідь на запит "Архітектура ПК" може бути виданий перелік статей, в яких згадується це поняття.
- Під **структурованими** даними розуміються правила, що визначають їх форму, розмір, значення. Структуровані дані зазвичай зберігаються в базах даних (БД).

База даних являє собою комп'ютерний варіант організованої інформації. Зазвичай елементи інформації об'єднує спільна тема або призначення, як, наприклад, список студентів з оцінками в журналі групи.

Прикладами великих інформаційних систем є банківські системи, системи замовлень залізничних квитків, бібліотека і т.д.

База даних – це сукупність даних і зв'язків між ними, великий масив різнорідної інформації, організованої за певними правилами. Ці правила передбачають загальні принципи опису, зберігання і обробки даних. Зазвичай база даних створюється для предметної області.

Предметна область – це частина реального світу, яка вивчається з метою створення бази даних. Набори принципів, за якими організована структура зберігання даних в базі, називаються моделями даних.

Існують 3 **основні моделі** даних – реляційні бази даних, ієрархічні і мережеві структури.

Ієрархічна модель дозволяє будувати БД з деревовидної структурою. У них кожен вузол містить свій тип даних (сутність). На верхньому рівні дерева є один вузол – корінь, на наступному рівні розташовуються вузли, пов'язані з цим коренем, потім вузли, пов'язані з вузлами попереднього рівня, і т. Д. Кожен вузол може мати тільки одного предка (Рис. 11.1).

11.2 Основні моделі даних

Існують 3 основні моделі даних – реляційні бази даних, ієрархічні і мережеві структури.

Ієрархічна модель даних

Ієрархічна модель дозволяє будувати БД з деревовидної структурою. У них кожен вузол містить свій тип даних (сутність). На верхньому рівні дерева є один вузол – корінь, на наступному рівні розташовуються вузли, пов'язані з цим коренем, потім вузли, пов'язані з вузлами попереднього рівня, і т. Д. Кожен вузол може мати тільки одного предка (Рис. 11.1).

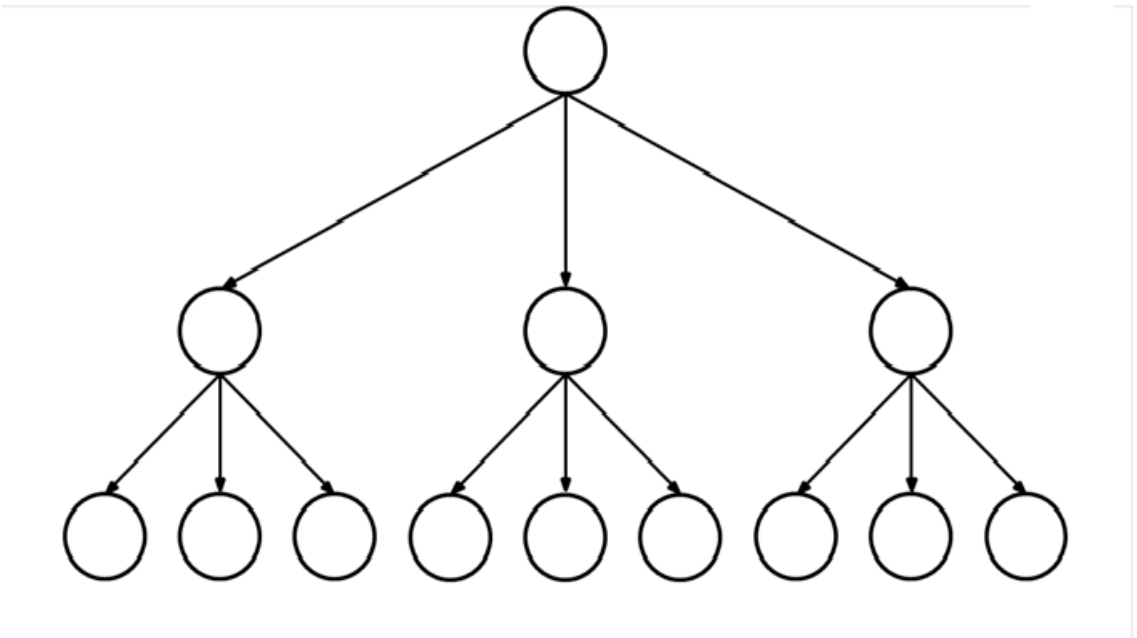


Рис 11.1 – Схема ієрархічної бази даних

Пошук даних завжди починається з кореня і йде у напрямку вниз, поки не знайдений потрібний об'єкт.

Переваги:

1) простота опису ієрархічних структур реального світу, 2) швидке виконання запитів, відповідних структурі даних.

Недоліки:

1) такі БД часто містять надлишкові дані,

2) не завжди зручно кожен раз починати пошук потрібних даних з кореня.

На Рис.11.2 можна побачити приклад ієрархічної моделі

Рис 11.2 Приклад ієрархічної моделі даних

Мережева модель

Мережева модель є розширенням ієрархічної. В ієрархічних структурах запис-нащадок повинен мати тільки одного предка; в мережевій структурі даних нащадок може мати будь-яке число предків (Рис. 11. 3).

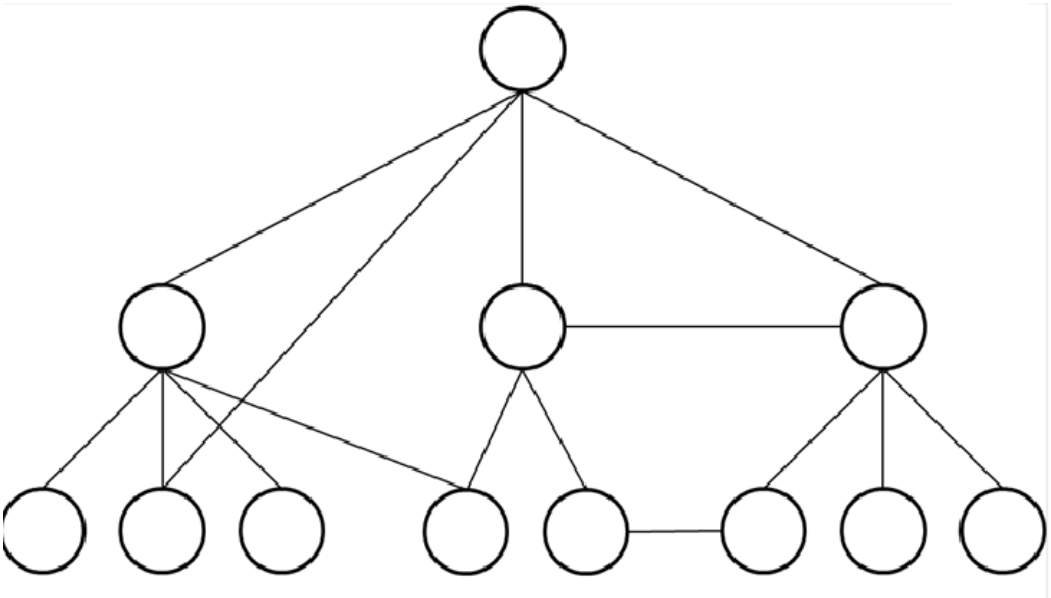


Рис. 11.3 Схема мережевої моделі даних

Переваги: ефективна реалізація за показниками витрат пам'яті та оперативності.

Недолік: висока складність і жорсткість схеми БД, побудованої на її основі.

Таким чином, в мережевий моделі можливі зв'язки всіх інформаційних об'єктів з усіма. Наприклад, кожен викладач може навчати багато студентів і кожен студент може навчатися у багатьох викладачів (Рис.11. 4).

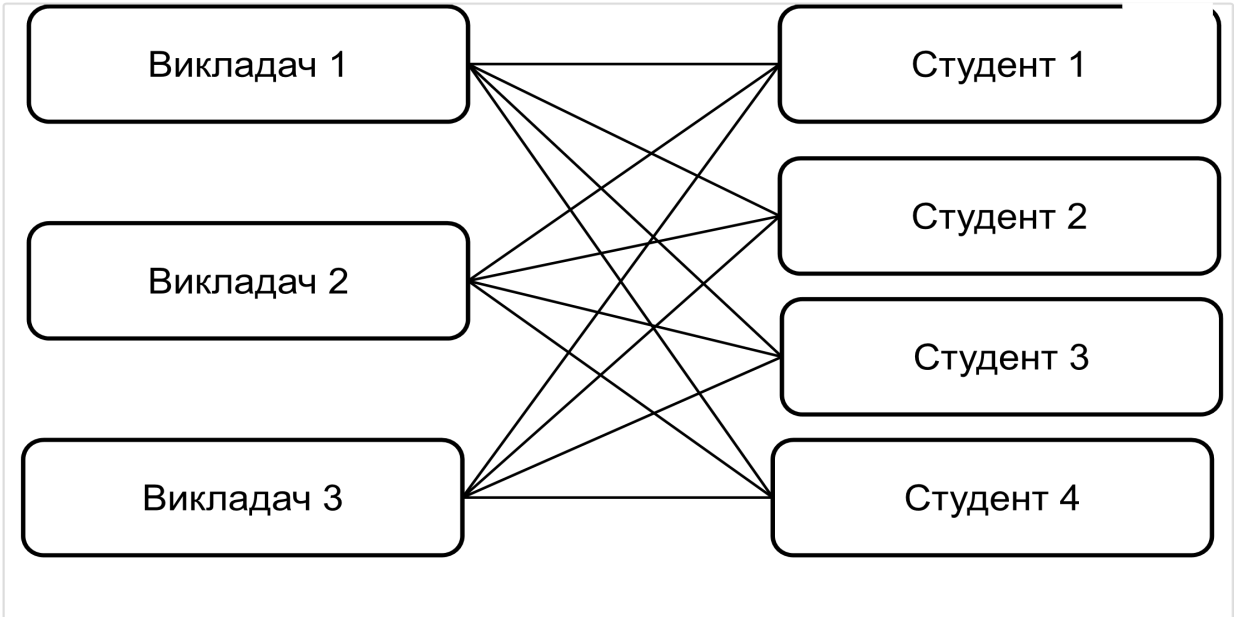


Рис. 11.4 – Приклад мережевої структури

Реляційна модель

Реляційна модель даних являє собою сховище даних, які організовані у вигляді двовимірних таблиць
(Рис. 11.5).

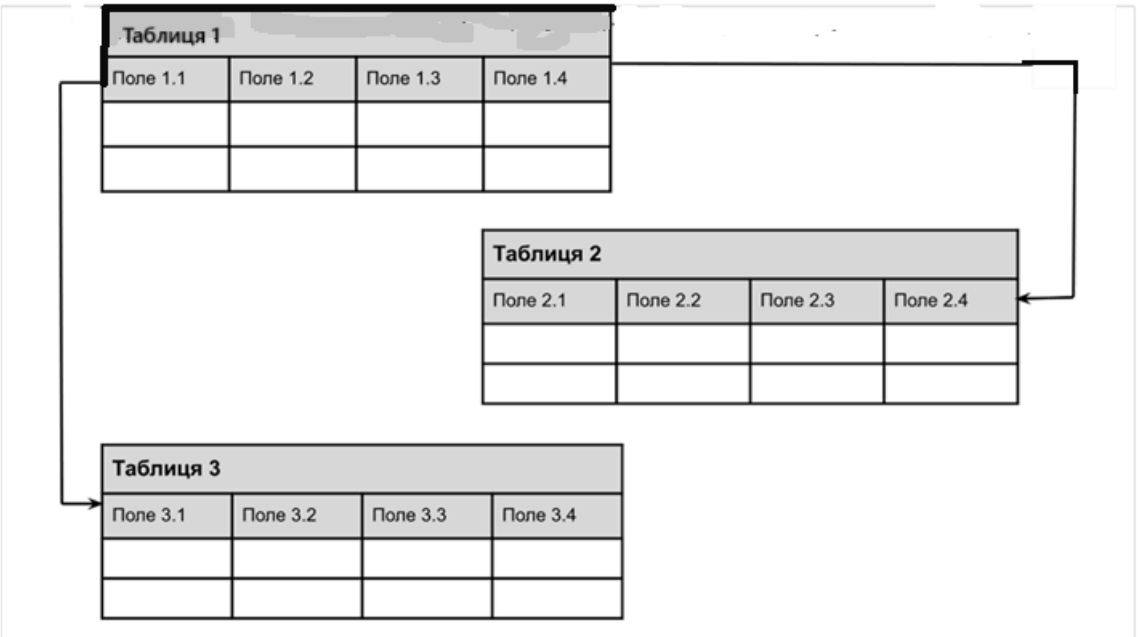


Рис. 11.5 Схема реляційної структури

Будь-яка таблиця реляційної бази даних складається з рядків, які називаються записами, і стовпців, які називаються полями. На перетині

стовпця і рядка знаходяться конкретні значення даних, що містяться в таблиці (Рис. 11.6).

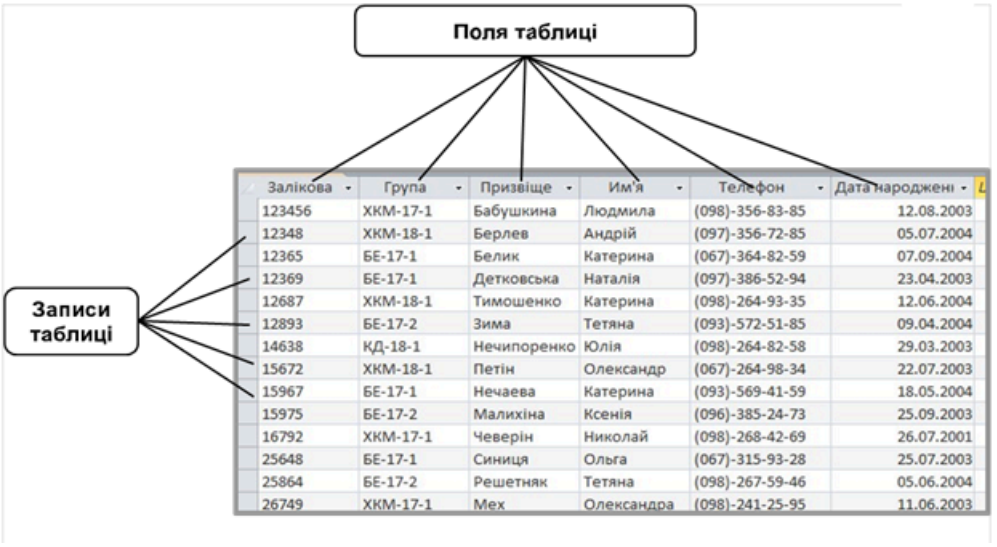


Рис. 11.6 – Таблиця бази даних

Таким чином:

Поле бази даних – це стовпець таблиці, що містить значення певної властивості.

Кожне поле характеризується своїм ім'ям і типом даних, що представляють значення даної властивості.

Запис бази даних – це рядок таблиці, що містить набір значень властивостей, розміщений в полях бази даних. Кожен запис являє собою набір значень, що містяться в полях.

Кожна таблиця повинна містити хоча б одне ключове поле, вміст якого унікальний для кожного запису в цій таблиці.

Ключове поле – це поле, значення якого однозначно визначає запис у таблиці.

Принципи організації даних:

1) Кожен елемент таблиці – один елемент даних.
2) Всі значення в одному стовпці є однорідними, тобто, мають один тип (числа, текст, дата і т. Д.).
3) Кожен запис у таблиці унікальний, тобто, в таблиці немає однакових рядків.
4) Кожне поле має унікальне ім'я.
5) Послідовність полів у таблиці несуттєва.
6) Послідовність записів у таблиці несуттєва.

11.3 СКБД Microsoft Access

Access – це система управління базами даних (СКБД). Вона призначена для зберігання і пошуку даних, подання інформації в зручному вигляді й

автоматизації часто повторюваних операцій (таких, як ведення рахунків, облік, планування тощо).

СУБД Access призначена для створення реляційних баз даних. Створення БД починається з проектування бази даних, тобто опису предметної області. Опис предметної області має охоплювати весь клас сутностей, інформація про яких повинна зберігатися в БД, і забезпечувати вимоги до функцій системи.

Система Access – це набір інструментів кінцевого користувача для управління базами даних. До її складу входять конструктори таблиць, форм, запитів і звітів.

База даних – це комп'ютерний термін, використовуваний для позначення інформації з певної теми або відомостей, пов'язаних з деяким додатком. Зберігання інформації у вигляді бази даних полегшує доступ до неї, пошук та вилучення потрібних фрагментів.

В Access база даних – це загальне сховище даних і відповідних їм об'єктів. **Об'єкти бази даних** – це таблиці, запити, форми, звіти, макроси і модулі (рис.11.7).

Ці об'єкти включають дані та інструментальні засоби, необхідні для використання Access.

Таблиця. Використовується для зберігання даних. Для відображення даних використовується

перегляд в режимі таблиці.

Запит. Дозволяє здійснювати пошук, сортування та витяг певних даних.

Форма. Забезпечує можливість введення і відображення даних в заданому форматі.

Звіт. Дозволяє відображати і друкувати відформатовані дані, включаючи результати обчислень і підсумкові значення.

Макрос. Включає прості команди для автоматизації виконання завдань без програмування.

Модуль. Програма, написана мовою Visual Basic for Application (VBA).

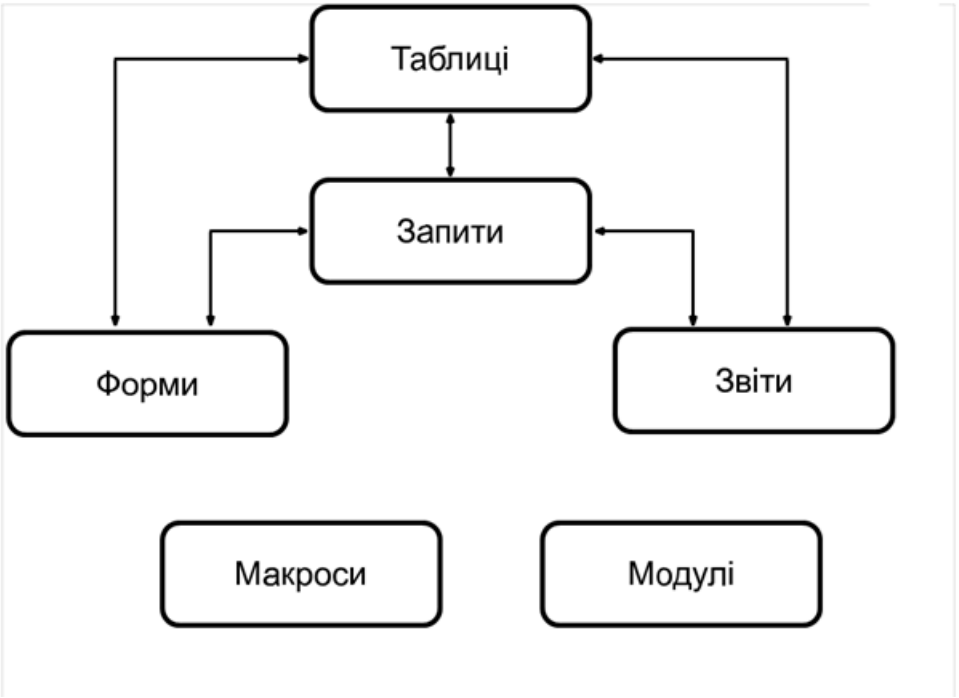


Рис. 11.7 Комп’ютерна система бази даних

Якщо уявити собі велику канцелярську шафу, повну всіляких папок, окремих листів і взагалі паперових обривків із записами будь-якого характеру то цю шафу можна цілком назвати повноцінною базою даних. Більш того, в доком’ютерну еру тільки так воно і було. Однак сама по собі ця шафа ніякої практичної цінності не має. У всякому разі, без кваліфікованого служителя, який би знав, що в шафі лежить, де конкретно воно знаходиться і як швидко можна отримати повну відповідь на питання, що цікавить. Так от, цей служитель укупі з обов'язковою системою організації папок в шафі і є системою управління базою даних, тобто СУБД.

У програмі MS Access вона виглядає наступним чином. **Вся вихідна інформація зберігається в чітко визначених таблицях.** Під чітким визначенням мається на увазі така структура таблиці, в якій кожен рядок має унікальний ідентифікатор (наприклад, номер рядка), а дані представлені стовпцями.

Таким чином, будь-яка таблиця є одновимірним набором записів. Неодмінною правилом створення таблиці в СУБД є строге визначення вмісту самої таблиці. В її осередках може зберігатися тільки фактична і тільки незмінна інформація. Це може здатися дещо дивним і незвичним для користувачів електронних таблиць, проте ні в MS Access, ні в СУБД взагалі, в осередках базових таблиць принципово не може бути обчислюваних значень.

Це не дуже зручно, проте, на щастя, існує можливість обійти цю заборону. Для цього в СУБД використовуються так звані запити. Власне кажучи, **запити – це ті ж самі таблиці, тільки вони заповнюються не вручну, а за допомогою заздалегідь заданих формул та інших залежностей.** Таким чином, те, що не можна в таблиці, можна в запиті: складати, віднімати, ділити, виконувати інші математичні або логічні операції, вибирати дані з якої-небудь умови. А свою назву запити отримали

від того, що вони схожі з широко поширеним природним дією аналогічного призначення. Прикладом запиту є така конструкція: хто конкретно купував автомобіль ВАЗ 2109 вишневого кольору в період з січня по грудень 1998 року і розплачувався при цьому готівковою іноземною валютою, в якості якої використовувалися швейцарські франки. Зіткнувшись з подібним запитом, СУБД самостійно перегляне відповідні таблиці, в яких зберігаються фактичні дані, і відбере з них всі рядки, які відповідають вимогам запиту. Причому з самими таблицями нічого не відбувається, в таблицю запиту передаються лише копії цих записів.

Запити стали основним робочим інструментом СУБД, завдяки тому, що в них поєднуються як можливості вибірки інформації з деякого її масиву, так і можливості зміни цієї інформації за допомогою формул. Одночасно з відбором, запити можуть виробляти будь-які розрахунки (наприклад, не тільки показати всіх покупців вишневих дев'яток, але і вказати, скільки кожна продаж принесла доходу, яка виявилася її собівартість, скільки довелося віддати в казну, а скільки залишилося в якості чистого прибутку) і навіть підбивати підсумки. В той же час, формально запити самі можуть бути представлені як таблиці і використані в подальшому в якості джерела даних для інших запитів. Це називається запит за запитом.

Само собою зрозуміло, що запити і таблиці лише тоді будуть жити і трудитися в мирі та злагоді, коли між ними з'являться однозначні і строго певні зв'язки. Ці зв'язки служать основою, на яку спирається вся система СУБД.

Теоретично, можна сконструювати таку загальну таблицю, в рамках якої можна уявити все зберігаються в базі дані, проте подібна таблиця виявляється майже повсюдно, по-перше, зайво громіздкою, а по-друге, заповненої повторюваними даними. Через громіздкість нею важко оперувати, а зайві дані збільшують її загальний обсяг, що обертається швидким зростанням вимог до системних ресурсів комп'ютера.

Припустимо, база даних містить інформацію про продажі. Отже, в ній неодмінно стануть присутнім стовпці з даними на самого клієнта. Наприклад, його прізвище, посада, телефон та адресу доставки. А тепер уявімо, що у нашої компанії з'явився постійний клієнт, який зробив протягом року п'ятдесят різнойменних покупок. Згідно з правилами організації таблиці, ці дані будуть повторюватися в кожній її рядку, а значить – вони займуть в сорок дев'ять разів більше місця, ніж їм насправді потрібно. У теж час, застосувавши систему зв'язків, можна створити окремо таблицю для оформлення замовлення і окремо таблицю з реєстром клієнтів. У тому випадку, коли покупець звернувся вперше, реєстр поповнюється новим рядком. Якщо ж він прийшов повторно, то, замість нового запису в реєстрі, в таблиці замовлення ставиться покажчик на вже існуючу рядок реєстру.

Важливою складовою СУБД є форми. Як відомо, існує велика різниця між тим, що виводиться на екран, і тим, що насправді записано на вінчестері. Приміром, на екрані ми бачимо зображення людини, а на жорсткому диску "лежать" тільки певним чином чергуються нулі і одиниці. Так от, ця фактично

зберігається інформація є прикладом фактичних даних, а відображається на її основі картинка – формою відображення. Причому від того, наскільки вірно організована форма, залежить, наскільки добре буде сприйнята або введена користувачем сама інформація. Всі знають, як важко буває заповнити навіть елементарний друкований бланк, якщо його поля забезпечені малозрозумілими підписами, та ще розкиданими в довільному порядку по листу.

Таким чином, форми в СУБД служать для зручного представлення інформації виключно для людських очей (наприклад, на паперовій роздруківці або частіше на екрані монітора).

Звіти використовуються для представлення даних в зручному вигляді. Звіт можна вивести на екран або частіше роздрукувати на принтері. У ньому можна групувати і сортувати дані в будь-якому порядку.

Звіти дозволяють відображати і друкувати дані з будь-яким ступенем деталізації. У них можна отримати результати складних розрахунків, статистичних порівнянь, а також помістити в нього малюнки та діаграми.

Контрольні питання

1 Назвіть основну проблему традиційних способів збереження та обробки інформації.
2 Що таке автоматизовані інформаційні системи і які нові можливості вони надають?
3 Яка основна відмінність між неструктурованими та структурованими даними?
4 Де зазвичай зберігаються структуровані дані?
5 Дайте визначення поняттю "база даних".
6 Що таке "предметна область" у контексті баз даних?
7 Які три основні моделі даних згадуються у тексті?
8 Опишіть, яку структуру має ієрархічна модель даних.
9 Скільки предків може мати кожен вузол в ієрархічній моделі даних?
10 У чому полягає основна відмінність між ієрархічною та мережевою моделями даних щодо зв'язків між об'єктами?
11 Назвіть одну перевагу та один недолік ієрархічної моделі даних.
12 Як організовані дані в реляційній моделі даних?
13 Що таке "ключове поле" в реляційній базі даних і яка його функція?
14 Назвіть три об'єкти бази даних в Access та коротко поясніть призначення кожного.

Лекція 12. Робота з базами даних в електронних таблицях.

12.1 Робота з базами даних в електронних таблицях

База даних – це організована у спеціальний спосіб сукупність даних. Інформація у базах даних зберігається у рядках та колонках так само, як і в електронних таблицях.

Microsoft Excel дає змогу створювати бази даних безпосередньо на робочому аркуші. Водночас рядки називаються *записами*, стовпчики – *полями*, а сама база даних – *списком*.

В Microsoft Excel є набір засобів, що полегшують оброблення й аналіз даних, що містяться у списку. Наприклад, для отримання впорядкованої інформації можна впорядкувати список за одним, двома або трьома стовпчиками. За необхідності можна приховати частину інформації, використовуючи фільтрацію. Для підрахунку підсумків за групами інформації, зручно скористатися можливістю підбиття проміжних підсумків.

Зазвичай таблиця-список значно відрізняється від баз даних у фахових системах керування базами даних, але наявність спеціальних команд і функцій для керування такою базою істотно спрощує роботу й розширює можливості оброблення даних. Окрім зазначених можливостей, електронні таблиці підтримують обмін даними із системами керування базами даних, що дає можливість читання або зберігання даних у традиційних форматах класичних баз даних.

Організація списку

Щоб успішно використовувати усі окреслені можливості роботи з обліковою інформацією, дані мають бути введені у список за такими правилами:

1. Варто уникати створення більше ніж одного списку на аркуші.
2. Проектувати список потрібно так, щоб кожен стовпчик містив подібні (однотипні) дані.
3. Між списком та іншими даними аркуша необхідно залишати, щонайменше, один порожній рядок і один порожній стовпчик.
4. Перед внесенням змін до списку варто переконатися у тому, що усі приховані рядки і стовпчики відображені.
5. Потрібно створити *заголовки* стовпчиків у першому рядку списку. Оформлення заголовків стовпчиків списку має відрізнятися від оформлення рядків даних.
6. У списку не має бути повністю порожніх рядків і стовпчиків.
7. На початку та наприкінці комірки не має бути пробілів (вони впливають на пошук і сортування).
8. Якщо потрібно зазначити довге ім’я поля для стовпчика з «вузькими» даними, можна записати ім’я у кількох рядках однієї комірки або змінити орієнтацію тексту в комірці.

Електронна таблиця розпізнає списки автоматично. Перед виконанням дій зі списком достатньо активізувати будь-яку комірку всередині списку. Заголовки стовпців, уведені у першому рядку, не опрацьовуються, як інші

дані. Якщо перед обробленням списку виділити окрему його частину, то електронна таблиця вважатиме списком лише виділений діапазон комірок.

Перед початком заповнення таблиці потрібно належно відформатувати комірки полів (насамперед для дат, часу, числових значень). Список може також містити значення полів, отриманих у результаті обчислення формул та випадаючи списки. Для формування списку, який містить значення, котрі повторюються, можна використовувати функцію автозаповнення. Також можна створити перелік обмежень на дані та створити повідомлення для користувача, які висвітлюються під час введення нових даних та під час появи помилок у процесі заповнення таблиці.

Присвоєння імені діапазону комірок, що містить список, спрощує роботу зі списком, особливо коли він великий.

Сортування списків

Microsoft Excel дає змогу виконувати просте (за одним стовпчиком) і складне (за кількома стовпчиками) сортування даних у таблиці-списку.

Сортування можна виконувати за алфавітом (від А до Я або від Я до А), числами (від найменших до найбільших або навпаки), а також датами і часом (від старих до нових або від нових до старих). Можна також виконувати сортування за налаштованими списками (таким, що складаються з елементів «Великий», «Середній» і «Маленький») або за форматом, включаючи колір комірок, колір шрифту, а також за значками.

Для роботи з базами даних використовується вкладка стрічки **Дані**. Для сортування списку необхідно виділити стовпчик, у якому проводитиметься сортування, і скористатися командою **Дані** ⇨ **Сортувати**. Програма автоматично визначить розмір списку, рядок з іменами полів. Водночас відкриється діалогове вікно **Сортування** (рис. 12.1), де обирають заголовок стовпчика, за яким проводитиметься сортування, тип сортування (**Значення, Колір клітинок, Колір шрифту, Піктограма умовного форматування**), порядок, за яким необхідно провести сортування.

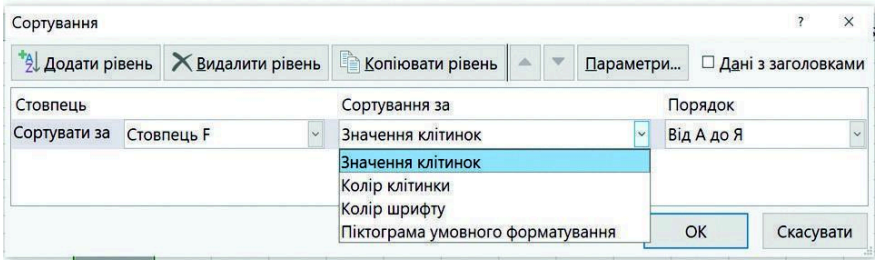


Рис. 12.1. Сортування даних

Текстовий процесор Microsoft Excel дає можливість сортувати список за багатьма стовпчиками одночасно. Для встановлення параметрів сортування за наступною колонкою натискають кнопку **Додати рівень**, а потім повторюють усі попередні кроки. Стовпці з вищою позицією у списку будуть відсортовані раніше, ніж стовпчики із нижчою позицією. Максимальне число стовпців для сортування – **64**.

Якщо результат сортування потрібно скасувати, можна скористатися комбінацією клавіш **Ctrl**+**Z**.

Крім того, доцільно зважати й на такі поради:

- *для швидкого відновлення початкового порядку сортування записів* у списку після застосування різноманітних складних сортувань можна до виконання цих дій створити додаткове поле з номерами записів і включити його до списку. Після цього, для відновлення початкового порядку достатньо відсортувати список за цим полем;
- *для сортування за значеннями тільки одного поля* можна також скористатися кнопками **Сортування від А до Я** та **Сортування від Я до А** вкладки **Дані**;
- *для сортування лише частини списку* перед звертанням до команд виділяють потрібний діапазон. Але зауважимо, що включення у діапазон виділення не усіх полів у записах списку (наприклад, список займає стовпчики від А до М, а виділено лише стовпці від С до Н) спричинить руйнацію внутрішніх зв'язків між даними у записах і список буде пошкоджено (наприклад, Іванов Петро може стати Іванов Марія).

За замовчуванням виконується сортування рядків списку. Для сортування даних за стовпцями потрібно у вікні **Сортування** натиснути кнопку **Параметри** та встановити перемикач **Стовпці діапазону**.

Під час сортування списків, що містять формули, потрібно дотримуватись простих правил, а саме:

- у формулах посилання на комірки (адреси), що належать до полів одного запису, оформлювати необхідно як відносні;
- посилання на комірки поза списком необхідно оформлювати у формулах лише як абсолютні;
- потрібно уникати у формулах посилань на комірки, що належать до полів з інших записів.

Фільтрування списків

Фільтрування – це швидкий і легкий спосіб пошуку підмножини даних у списку. У відфільтрованому списку відображаються лише рядки, що відповідають умовам відбору, які задані для стовпчика. На відміну від сортування, під час фільтрування порядок записів у списку не змінюється. Під час фільтрування тимчасово приховуються рядки, які не потрібно відображати.

Рядки, відібрані під час фільтруванні в Microsoft Excel, можна редагувати, форматувати і виводити на друк, також на їх основі можна створювати діаграми.

Для фільтрування даних в Microsoft Excel використовується два типи фільтрів: **Автофільтр** і **Розширений фільтр**. Під час використання **Автофільтру** фільтрування здійснюється безпосередньо на основі вихідних даних. Під час використання **Розширеного фільтру** є можливість розмістити результати фільтрування в окремій області.

Після фільтрування відібрані дані можна копіювати традиційними методами в інше місце робочої книги.

Якщо перед фільтруванням виділити частину списку, то можливості відбору будуть надані лише із записів і за полями, що входять у діапазон виділення.

Фільтрування даних за допомогою Автофільтра. За допомогою автофільтра можна створити фільтри трьох типів: *за списком значень*, *за форматом* або *за умовами*. Усі вони є взаємовиключними у межах діапазону комірок або стовпчика таблиці. Наприклад, можна виконати фільтрування за кольором комірок або за списком чисел, однак використовувати фільтри обох типів одночасно не можна.

Для відбору даних зі списку за допомогою фільтра необхідно: виділити будь-яку комірку в середині списку та на вкладці **Дані** натиснути кнопку **Фільтр**. У заголовках стовпців з'являться кнопки **Автофільтри**, які позначають, що фільтрація включена, але не застосована (рис. 12.2). Ці кнопки зі стрілкою, що з'являються справа від кожного імені поля, дають змогу відкрити перелік значень поля і накласти обмеження, створивши критерій відбору за кількома полями.

	A	B	C	D
1	Номер	Прізвище	Дата народжен	Місце проживані
2	1	Адамчук А.В.	15.04.1995	Львів
3	2	Бичинюк О.С.	21.02.1980	Одеса
4	3	Іванов К.Л.	05.05.1991	Львів
5	4	Баран О.С.	27.05.1973	Дніпро
6	5	Проць О.Д.	03.03.1985	Львів
7	6	Кмить А.Д.	08.12.1951	Одеса

Автофільтр

Рис. 12.2. Автофільтр

Список із запропонованими для вибору значеннями (рис. 12.3) може бути довгий. Для швидкого переходу до певного елемента списку можна ввести початкові літери у поле **Пошук**.

Елемент **Виділити все** списку дає змогу зняти обмеження щодо поля.

Якщо у стовпчику містяться текстові дані, вибирають команду **Текстові фільтри**. Якщо у стовпчику містяться числа, вибирають команду **Числові фільтри**. Якщо стовпець містить дати – **Фільтри за датою**.

Після вибору відповідного критерію фільтрування (**Закінчується**…; **Містить**...; Не **дорівнює**... тощо). Microsoft Excel відобразить вікно **Користувацький автофільтр** (рис. 12.4), у якому вибираються оператори порівняння, на основі яких буде проведено відбір даних.

Якщо необхідно задати другу умову відбору даних, вибирають або **І** (якщо потрібно, щоб обидва критерії порівняння застосовувалися), або **АБО** (якщо потрібно застосувати один із критеріїв порівняння). Під час визначення критеріїв за текстовими полями використовуються також символи шаблона «?» та «*». Застосувати автофільтр можна лише до одного списку на робочому аркуші.

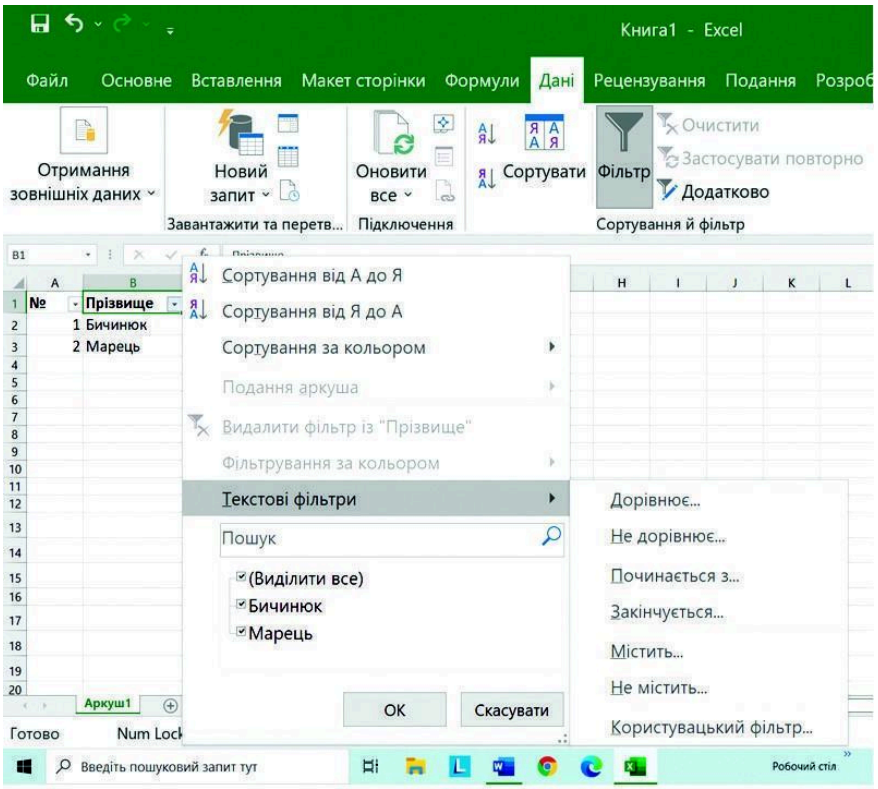


Рис. 12.3. Критерії фільтрування текстових даних

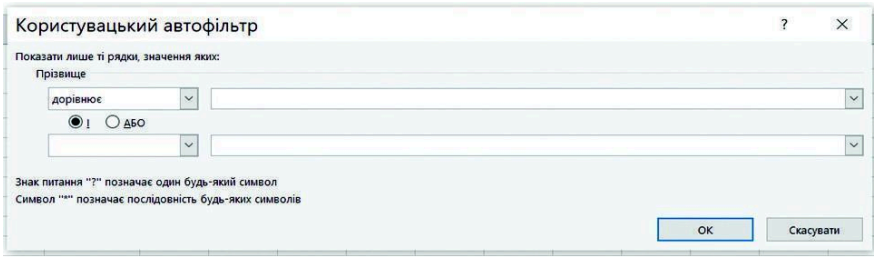


Рис. 12.4. Користувацький автофільтр

Фільтрування із застосуванням розширеного фільтра. Команда Розширений фільтр використовується для створення складних умов відбору даних і вимагає визначення *рядків вихідного списку, діапазону умов та діапазону результату* (рис. 12.5).

	A	B	C	D	E	F	G	H	I
1	№	Дата	Послуга	Ціна	Кількість	Сума		Дата	Послуга
2	1	26.09.2016	Реєстрація ТОВ	1350	5	6750		29.09.2016	З*
3	2	26.09.2016	Реєстрація ПП	2000	1	2000		29.09.2016	*я
4	3	28.09.2016	Реєстрація ФОП	700	2	1400		29.09.2016	М*ц*
5	4	29.09.2016	Зміна директора	1000	3	3000			
6	5	29.09.2016	Зміна складу засновників	1350	4	5400			
7	6	29.09.2016	Зміна назви	1600	5	8000			
8	7	05.10.2016	Зміна місцезнаходження	1400	6	8400			
9	8	29.09.2016	Ліквідація ФОП	3000	2	6000			
10	9	28.09.2016	Виписка з ЄДРПОУ	250	7	1750			
11	10	26.09.2016	Витяг з ЄДРПОУ	200	1	200			
12									
13	№	Дата	Послуга	Ціна	Кількість	Сума			
14	4	29.09.2016	Зміна директора	1000	3	3000			
15	5	29.09.2016	Зміна складу засновників	1350	4	5400			
16	6	29.09.2016	Зміна назви	1600	5	8000			
17	8	29.09.2016	Ліквідація ФОП	3000	2	6000			

Діапазон умов

Вихідний список

Діапазон результату

Рис. 12.5. Фільтрування із застосуванням розширеного фільтра

Для створення діапазону умов необхідно:

1. У вільні комірки поза списком скопіювати заголовки тих стовпчиків, на дані яких будуть накладені обмеження.

2. Під кожним із заголовків задати умови відбору даних.

Умови відбору даних можуть бути даними або їх елементами, формулами (наприклад, формулою, яка фільтрує усі значення у стовпчику, що перевищують середнє значення), можуть містити оператори порівняння та символи підстановки «?» або «*».

Задаючи текстові умови, слід дотримуватися таких правил:

- одна літера означає, що пошуку підлягають усі значення, що починаються із зазначеної літери;
- символ «>» або «<» означає, що пошуку підлягають значення, що за абеткою стоять після уведеного текстового значення або перед ним;
- формула *=текст* означає, що пошуку підлягають значення, які точно збігаються із введеним рядком символів *текст*.

Умови, що знаходяться в одному рядку, у Microsoft Excel розглядаються як об'єднані оператором **І**, а ті, що знаходяться у різних рядках, розглядаються як об'єднані оператором **АБО**.

Для створення діапазону результату необхідно:

1. У вільний рядок поза таблицею скопіювати рядок заголовків вихідного списку (діапазон результату може перебувати лише на тому ж робочому аркуші, на якому знаходиться вихідний діапазон).

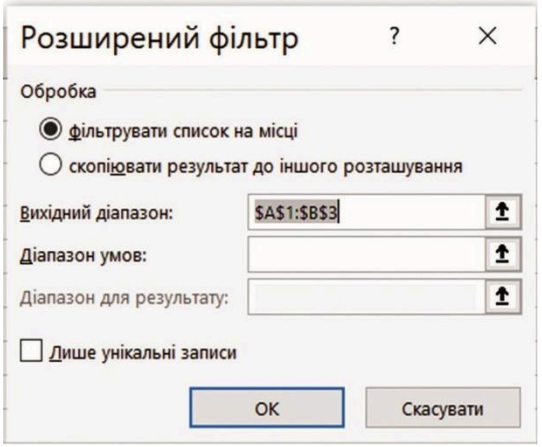


Рис. 12.6. Задання параметрів Розширеного фільтра

2. Виділити будь-яку комірку у середині списку.

3. Виконати послідовність команд: вкладка **Дані** група **Сортування й фільтр** кнопка **Додатково**.

4. У вікні **Розширений фільтр** (рис. 12.6) встановити такі параметри:

- щоб результати фільтрування відображалися безпосередньо у списку, із приховуванням рядків, що не відповідають критеріям відбору, потрібно вибрати варіант **фільтрувати список на місці**, а якщо результат відбору має поміщатися в діапазон результату, потрібно встановити перемикач **скопіювати результат до іншого розташування**;

– у полі **Вихідний діапазон** вказується діапазон комірок, що фільтруються (діапазон має містити усі комірки списку з урахуванням комірок заголовків стовпців);

– у полі **Діапазон умов** вказується діапазон сформованих раніше умов (він має містити усі комірки діапазону умов відбору з урахуванням комірок заголовків стовпців);

– у разі розміщення результатів фільтрування в діапазоні ре-зульт ату потрібно переконатися, що у полі **Діапазон для результату** містяться усі комірки цього діапазону.

Контрольні питання

1. Визначте, що таке база даних у Microsoft Excel та як організувати її у вигляді списку.
2. Поясніть, які правила потрібно враховувати під час створення списку в Excel.
3. Опишіть, як виконується сортування даних у таблицях Excel і які методи сортування існують.
4. Перерахуйте, які можливості надає фільтрація списків у Microsoft Excel.
5. Продемонструйте, як працює автофільтр у Microsoft Excel і які параметри він підтримує.
6. Роз’ясніть, як використовувати розширений фільтр для створення складних умов відбору даних.

Лекція 13. Проміжні підсумки та консолідація даних.

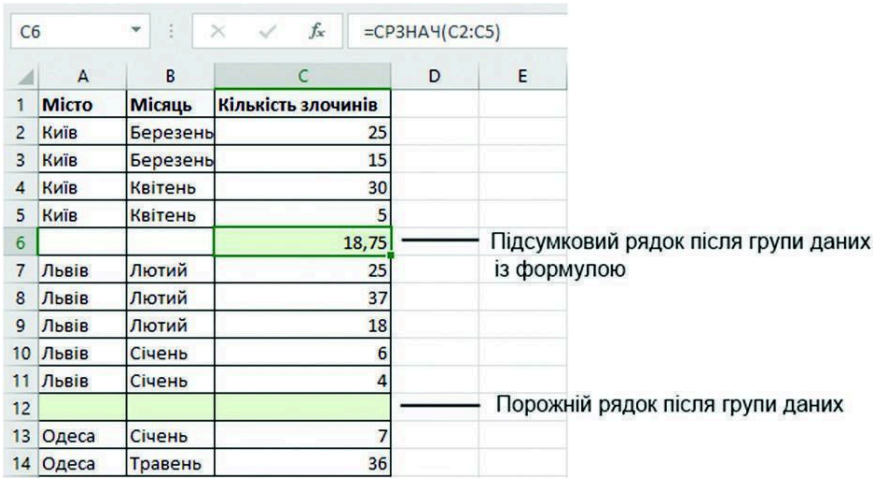
13.1 Робота із структурою даних

Структура даних здебільшого створюється для великих табл иць, що складаються з сотень рядків і (або) стовпців, і містять дані, які логічно можуть бути об’єднані в групи. Її можна створювати з рядків, стовпців або з рядків і стовпців одночасно.

Існує два способи структурування даних на робочому аркуші – автоматично та вручну.

Автоматичне створення багаторівневої структури. Для автоматичного структурування даних робочого аркуша необхідно:

1. Переконатися, що перший рядок (стовпчик) містить заголовки, кожен рядок (стовпчик) містить дані одного типу, в діапазоні даних відсутні порожні рядки або стовпчики.
2. Відсортувати рядки (стовпчики) для формування груп даних.
3. Вставити підсумкові рядки (стовпчики) з формулами після кожної групи даних (рис.13.1).
4. На вкладці **Дані** у групі **Структура** натиснути стрілку кнопки **Групувати** і виконати однойменну команду.



	A	B	C	D	E
1	**Місто**	**Місяць**	**Кількість злочинів**		
2	Київ	Березень	25		
3	Київ	Березень	15		
4	Київ	Квітень	30		
5	Київ	Квітень	5		
6			18,75		
7	Львів	Лютий	25		
8	Львів	Лютий	37		
9	Львів	Лютий	18		
10	Львів	Січень	6		
11	Львів	Січень	4		
12					
13	Одеса	Січень	7		
14	Одеса	Травень	36		

Рис. 13.1. Вставлення підсумкових рядків

Групування даних у ручному режимі. Для структурування робочого аркуша у ручному режимі необхідно:

1. Відсортувати рядки (стовпчики) для формування груп даних.
2. Вставити підсумкові рядки (стовпчики) з формулами після кожної групи даних або розділити групи порожніми рядками (стовпчиками) (див. рис. 13.1).
3. Виділити усі рядки (стовпчики), які мають входити в одну групу (не включати в діапазон даних підсумковий або порожній рядок, що використовується для поділу груп даних).
4. На вкладці **Дані** у групі **Структура** натиснути кнопку **Групувати**. Поруч з групою з'являться знаки структури.
5. Продовжувати виділення і групування внутрішніх рядків (стовпчиків) доти, доки не будуть створені усі необхідні рівні структури.

За необхідності, групування окремих елементів структури можна скасувати. Для цього необхідно виділити ті елементи багаторівневої структури, які мають бути виключені з групи та виконати послідовність команд **Дані** ⇨ **Структура** ⇨ **Розгрупувати**…

13.2 Підбиття підсумків

Для визначення сумарних, середніх, тих чи інших показників діяльності установ чи її підрозділів у різних у різні проміжки часу можна скористатися функцією автоматичного підведення підсумків у середині таблиці.

Для отримання таких проміжних підсумків необхідно виконати такі кроки:

1. Провести сортування за тим стовпчиком, за яким необхідно підбити підсумки з метою групування даних.
2. На вкладці **Дані** у групі **Структура** натиснути кнопку **Проміжні підсумки**.

3. У вікні **Проміжні підсумки** (рис. 13.2) у полі При кожній зміні в: вибрати стовпчик, за яким відбувалося сортування даних.

4. У полі **Використовувати функцію** вибрати функцію для розрахунку підсумкових значень: **Сума, Кількість, Середнє, Максимум, Мінімум** тощо.

5. У списку **Додати підсумки** до встановити прапорці біля найменувань тих стовпчиків, за якими проводитимуться обчислення.

6. У разі необхідності знімаються або встановлюються прапорці таких параметрів:

– **замінити поточні підсумки** – обчислення нових проміжних підсумків для заміни поточних;

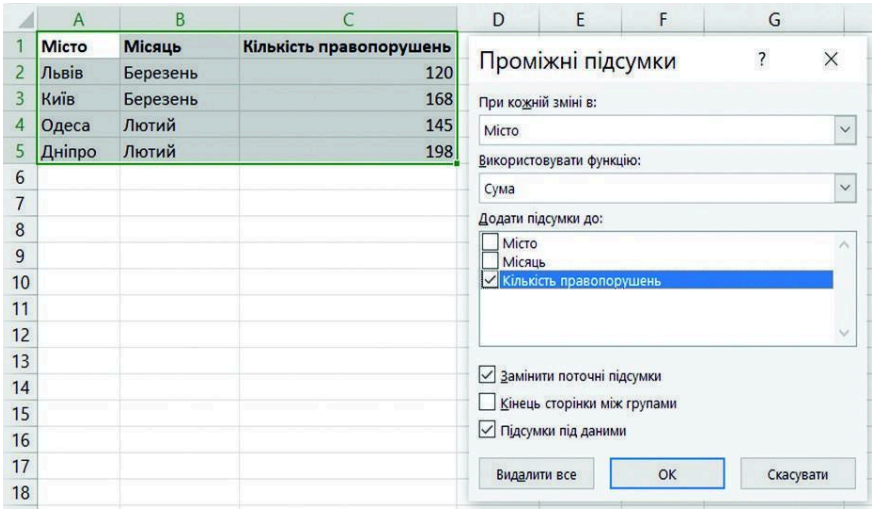


Рис. 13.2. Вікно Проміжні підсумки

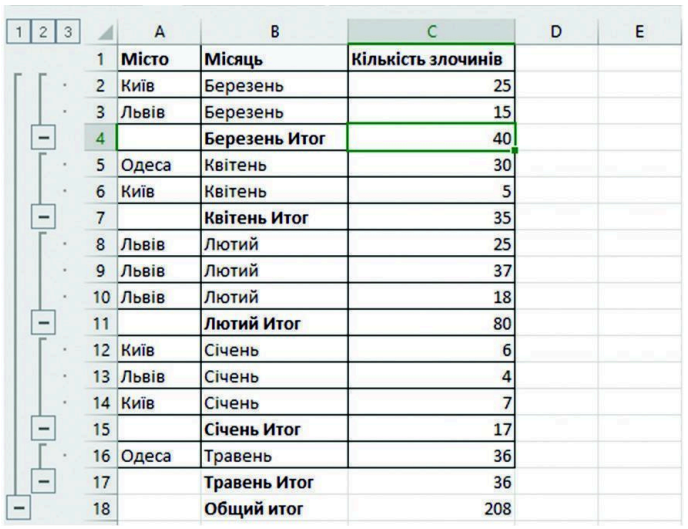
– **кінець сторінки між групами** – вставлення автоматичних розривів сторінок після кожної групи проміжних підсумків;

– **підсумки під даними** – вставлення рядків проміжних підсумків і загальних підсумків під позиціями даних.

Кнопка **Видалити все** видаляє усі проміжні підсумки.

До вже обробленої таблиці можна вдруге застосувати проміжні підсумки.

Операція автоматичного підбиття проміжних підсумків в Microsoft Excel не лише здійснює обчислення підсумкових сум і їх занесення на робочий аркуш, а й одночасно створює на робочому аркуші структуру, пов’язану з отриманими результатами (рис. 13.3).



	A	B	C	D	E
1	**Місто**	**Місяць**	**Кількість злочинів**		
2	Київ	Березень	25		
3	Львів	Березень	15		
4		**Березень Итог**	40		
5	Одеса	Квітень	30		
6	Київ	Квітень	5		
7		**Квітень Итог**	35		
8	Львів	Лютий	25		
9	Львів	Лютий	37		
10	Львів	Лютий	18		
11		**Лютий Итог**	80		
12	Київ	Січень	6		
13	Львів	Січень	4		
14	Київ	Січень	7		
15		**Січень Итог**	17		
16	Одеса	Травень	36		
17		**Травень Итог**	36		
18		**Общий итог**	208		

Рис. 13.3. Структура даних із підведенням проміжних підсумків

13.3 Приховування або відображення структурованих даних

Після створення структури на аркуші з'являться кнопки керування структурою:

1 2 3 – рівні організації даних. Взагалі може бути до 8 рівнів вкладеності. Натискаючи відповідні кнопки можна представити таблицю у потрібному рівні деталізації. Рівень 1 – показ тільки загального підсумкового рядку або стовпця. Рівень 2 – показ проміжних підсумків. Останній рівень – показати усі дані.

– приховування даних групи

– відображення даних групи

Якщо на аркуші відсутні символи структури, необхідно на вкладці Файл вибрати команду **Параметри** та у розділі **Додатково** в групі **Пара метри відображення цього аркуша** вибрати аркуш, на якому розміщуються структуровані дані та встановити прапорець **Відображати позначки структури в разі її застосування**.

13.4 Консолідація даних

Щоб підбити підсумки і скласти звіт за однотипними даними, які розташовані на декількох аркушах, можна консолідувати дані з окремих аркушів на один основний. Окремі аркуші можуть перебувати у тій же книзі, що й основний аркуш, або в інших книгах. Під час консолідації даних вони компонуються так, що їх стає простіше оновлювати та узагальнювати на регулярній основі або за необхідності.

Консолідація – це об'єднання даних з різних таблиць, розташованих у різних місцях, які мають однотипні дані в одну таблицю.

Є два основних способи консолідації даних:

1. **Консолідація за розташуванням.** Цей метод використовується, якщо дані з різних джерел впорядковані однаково і в них використовуються одні й ті ж підписи стовпчиків і рядків (наприклад, за наявності декількох аркушів витрат, створених з одного і того ж шаблону).

2 **Консолідація за категоріями.** Цей метод використовується, якщо дані з різних джерел впорядковані по-різному, але у них використовуються одні й ті ж підписи стовпчиків і рядків (наприклад, якщо дані за видами злочинів на різних аркушах знаходяться у різних рядках таблиці, містять різні елементи або різну кількість елементів, але макет таблиці однаковий на усіх аркушах).

Консолідація даних за розташуванням

Для проведення консолідації даних за розташуванням необхідно:

1. На кожному аркуші, який потрібно консолідувати, підготувати дані таким способом:

- усі діапазони даних мають були представлені у форматі списку:

перший рядок кожного стовпчика містить підпис, інші рядки – однотипні дані; порожні рядки або стовпчики у списку відсутні;

- кожен діапазон має розташовуватися на окремому аркуші (на ар-куші, на якому має виконуватися консолідація, діапазони не розташовуються);
- макети усіх діапазонів мають збігатися.

2. На основному аркуші виділити комірку, починаючи з якої будуть розташовуватися консолідовані дані. Переконатися, що справа і знизу цієї комірки достатньо вільних комірок, щоб не перезаписати наявні дані консолідованими даними.

3. На вкладці Дані у групі Знаряддя даних вибирають кнопку Консолідація.

4. У вікні Консолідація (рис. 13.4) у списку Функція вибрати функцію, яка буде використовуватися для консолідації даних (Сума, Кількість, Середнє, Максимум тощо).

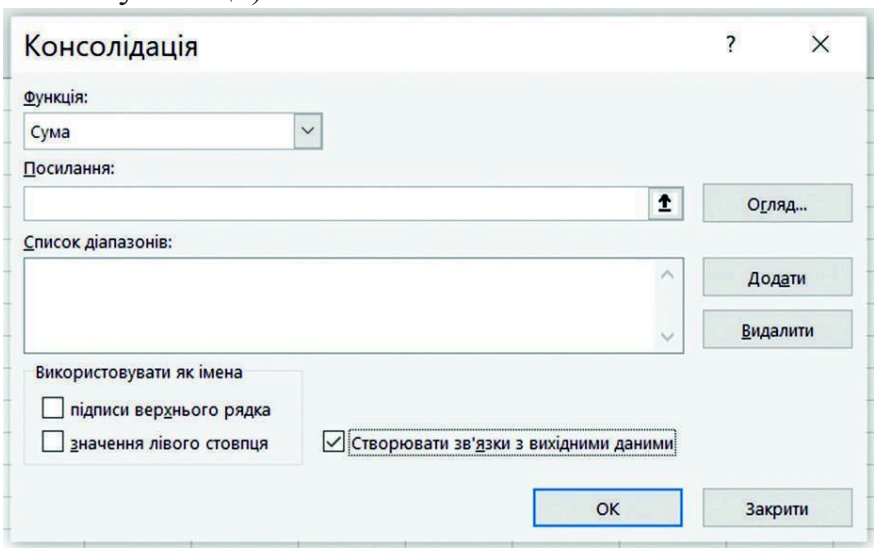


Рис. 13.4. Консолідація даних за розташуванням

5. У полі Посилання обирають або діапазон даних на одному з аркушів (якщо дані консолідації знаходяться у тій же книзі) або задають шлях до іншої книги ([КнигаN]ЛистN!Діапазон консолідованих комірок). Натискають кнопку Додати. Microsoft Excel скопіює посилання з поля Посилання у поле Список діапазонів.

6. Дію 5 повторюють доти, доки не додадуть усі потрібні діапазони з усіх аркушів.

7. Якщо консолідація має оновлюватися автоматично під час зміни вихідних даних в іншій книзі, встановлюється прапорець у полі Створювати зв'язки з вихідними даними.

Консолідація даних за категоріями

Спосіб консолідації даних за категоріями використовують як основу для об'єднання заголовки стовпчиків або рядків. Цей спосіб є гнучкішим. Наприклад, якщо дані за певним видом правоп орушень на різних аркушах знаходяться у різних рядках або стовпчиках таблиці (рис. 13.5), то під час консолідації за категоріями такі дані будуть об'єднані (рис. 13.6).

Рис. 13.5. Вхідні дані із різною структурою діапазонів, які використовуються для консолідації

	A	B	C
1	Види злочинів		
2	Особливо тяжких	365	
3	Тяжких	2878	
4	Середньої тяжкості	4071	
5			
6			
7			

Лист1 | Лист2 | Лист4

Рис. 13.6. Результат консолідації даних за категоріями

Для проведення консолідації даних за категоріями необхідно:

1. Представити усі діапазони даних у форматі списку та розташувати кожен діапазон на окремому аркуші.
2. На основному аркуші виділити комірку, починаючи з якої будуть розташовуватися консолідовані дані.

3. На вкладці **Дані** у групі **Знаряддя даних** вибирають кнопку **Консолідація** та задати усі діапазони консолідованих даних.

4. У вікні **Консолідація** (див. рис. 13.4) вибрати у списку **Функція** функцію, яка буде використовуватися для консолідації даних.

5. У розділі **Використовувати як імена** встановити такі прапорці:

- **підписи верхнього рядка** – якщо верхній рядок кожного з виділених діапазонів містить підписи стовпчиків;
- **значення лівого стовбця** – якщо лівий стовпчик кожного з виділених діапазонів містить підписи рядків.

6. Встановити прапорець **Створювати зв'язки з вихідними даними**, якщо необхідно, щоб під час зміни вихідних значень змінювалися значення у основній таблиці.

7. Натиснути **ОК**.

Усі назви, що не збігаються із назвами в інших вихідних областях, призведуть до появи додаткових рядків або стовпчиків у консолідованих даних.

Усі категорії, які не потрібно консолідувати, мають мати унікальні підписи, які трапляються лише в одному діапазоні вихідних даних.

Розглянемо приклад. Нехай робоча книга містить листи, які зберігають інформацію про нарахування заробітної плати працівникам протягом року. Для того щоб визначити розмір відпускних, необхідно встановити, яка середня заробітна плата кожного працівника.

Для цього потрібно об'єднати таблиці з інформацією про кожний місяць в одну. При цьому для кожного працівника необхідно визначити середню суму нарахувань.

Додаємо до робочої книги лист Відпускні. Скористаємося меню Дані - Консолідація. У вікні Консолідація задаємо перелік діапазонів консолідації та параметри консолідації.

Для того щоб додати діапазон даних, необхідно у вікні поля Посилання натиснути кнопку Огляд, вибрати необхідний діапазон даних на одному з листів (виділити дані разом із заголовками рядків та стовпців), натиснути у вікні Консолідація кнопку Додати.

Після натискання ОК з'явиться таблиця консолідації, яка буде містити зведену інформацію.

У лівій частині вікна відображаються кнопки, аналогічні кнопкам при побудові підсумків. Вони використовуються, щоб переглянути інформацію, на основі якої побудована таблиця консолідації.

Якщо необхідно змінити функцію консолідації, потрібно скористатися меню Дані – Консолідація і у вікні Консолідація зі списку функцій вибрати потрібну (Середнє).

При консолідації даних необхідно вибрати місце для розміщення підсумкового звіту, функцію (наприклад, *сума*) і джерела даних для консолідації.

Підсумковий звіт можна розмістити на одному листі з початковими даними, на іншому листі тієї самої книги або взагалі в іншому файлі.

Функція, яка використовується в консолідації, залежить від типу даних і виду звіту, який складається.

У разі зміни початкових даних консолідацію необхідно повторити. Якщо структура початкових таблиць діапазонів не міняється, то постійного повторення цієї процедури можна уникнути шляхом зв'язування консолідованих даних з початковими (відповідний перемикач).

Щоб установити зв'язок між консолідованими і початковими даними, необхідно при виконанні консолідації увімкнути опцію

Створювати зв'язки з вихідними даними в діалоговому вікні *Консолідація.*

У результаті активізації зазначеної опції між початковими даними і результатами консолідації буде встановлено динамічний зв'язок, який автоматично оновлює дані.

Якщо таблиця даних має заголовки рядків і стовпців, це теж необхідно вказати при консолідації (за допомогою перемикачів у вікні *Консолідація*).

Контрольні питання.

1. Поясніть, які правила потрібно враховувати під час створення списку в Excel.
2. Опишіть, як виконується сортування даних у таблицях Excel і які методи сортування існують.
3. Перерахуйте, які можливості надає фільтрація списків у Microsoft Excel.
4. Продемонструйте, як працює автофільтр у Microsoft Excel і які параметри він підтримує.
5. Роз'ясніть, як використовувати розширений фільтр для створення складних умов відбору даних.
6. Поясніть, які функції обчислення можна застосовувати у зведених таблицях.
7. Покажіть, як виконати консолідацію даних у Microsoft Excel та налаштувати її параметри.

Лекція 14. Зведені таблиці як інструмент аналізу даних.

Для ефективнішого аналізу даних у великих таблицях та таблицях, які мають різні повторення значень у стовпчиках та рядках, в електронних таблицях використовується такий інструмент, як зведені таблиці.

Зведена таблиця – таблиця, що забезпечує фільтрування даних за обраними стовпчиками та підбиття проміжних підсумків для більш зручного аналізу великих обсягів даних і прийняття обґрунтованіших рішень. Вони представляють великий обсяг інформації у стислому та зручному вигляді і надають можливість вирахувати підсумкову інформацію, не використавши жодної формули і не скопіювавши жодної комірки.

Підготовка вихідної таблиці

Для створення зведеної таблиці необхідно мати вихідну таблицю-список (рис. 14.1), яка має задовольняти таким вимогам:

	A	B	C	D	E
1	Прізвище	Вид послуги	Вартість послуги	Кількість годин роботи	Дата
2	Кіт А.	Завірення документів	250	2	05.03.2016
3	Семенів В.	Консультація	300	3	05.03.2016
4	Палига Ю.	Реєстрація підприємства	260	1	07.03.2016
5	Мних С.	Реєстрація підприємства	75	5	07.03.2016
6	Кіт А.	Завірення документів	100	4	07.03.2016
7	Семенів В.	Завірення документів	400	2	10.03.2016
8	Палига Ю.	Консультація	425	1	11.03.2016
9	Семенів В.	Реєстрація підприємства	456	3	12.03.2016
10	Кіт А.	Завірення документів	50	8	12.03.2016
11	Палига Ю.	Консультація	150	5	12.03.2016
12	Палига Ю.	Консультація	200	4	12.03.2016
13	Мних С.	Завірення документів	220	2	16.03.2016

Рис. 14.1. Приклад вихідної таблиці

- кожен стовпчик має мати заголовок;
- хоча б один стовпчик має містити повторювані значення;
- хоча б один стовпчик має містити числову інформацію, яка буде використовуватися для створення проміжних підсумків; – у кожен стовпчик мають вводитися дані в одному форматі; – у таблиці не має бути повністю незаповнених рядків та стовпців.

Створення рекомендованої зведеної таблиці

Рекомендовану зведену таблицю створюють у разі, коли бракує досвіду роботи. Під час використання цієї функції Microsoft Excel визначає структуру таблиці, зіставляючи дані з найвідповіднішими областями зведеної таблиці. Створивши просту зведену таблицю, можна змінити її орієнтацію та впорядкувати поля іншим чином, щоб досягнути кращих результатів.

Для створення рекомендованої зведеної таблиці необхідно:

1. Виділити комірку в діапазоні, що містить вхідні дані, та на вкладці Вставлення виконати команду **Таблиці** ⇨ пункт **Рекомендовані зведені таблиці**.
2. Із запропонованих варіантів зведених таблиць вибрати такий, що найбільше підходить.

Створення зведеної таблиці вручну

Якщо користувач знає як потрібно впорядкувати дані, можна створити зведену таблицю вручну.

Для цього необхідно:

1. Виділити комірку у діапазоні, що містить вхідні дані та на вкладці **Вставлення** вибрати команду **Таблиці** ⇨ **Зведена таблиця**.
2. У вікні **Зведена таблиця з таблиці або діапазону** (рис. 14.2) перевірити у полі **Таблиця або діапазон** діапазон комірок, що містить базові

дані та задати розташування на: **Новий** або **Наявний аркуш**. Натиснути **ОК**.

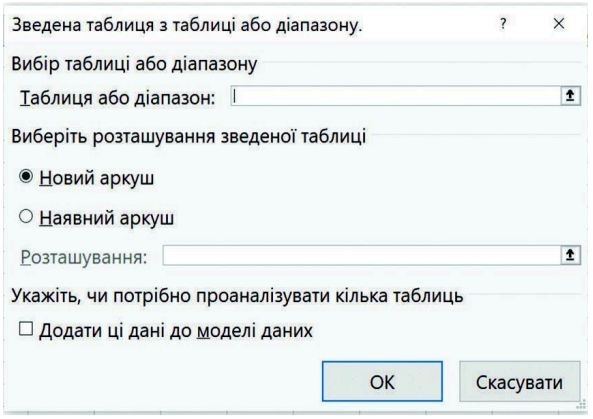


Рис. 14.2. Вікно Зведена таблиця з таблиці або діапазону

3. Microsoft Excel вставить порожній звіт зведеної таблиці у вказане місце і відкриє список полів зведеної таблиці, за допомогою якого можна:

– додавати та видаляти поля, встановивши або знявши прапорець для поля в області **Виберіть поля, які слід додати до звіту** (рис. 14.3). За замовчуванням нечислові поля додаються до області **Рядки**, ієрархії дати й часу – до області **Стовпці**, а числові поля – до області **Значення**;

– переміщувати поля, шляхом перетягування полів з однієї області у списку полів зведеної таблиці до іншої, наприклад, з області **Рядки** до області **Стовбці**.

Обчислення у зведених таблицях

Microsoft Excel автоматично визначає функцію, на основі якої проводяться підсумкові обчислення. Здебільшого програма застосовує операцію сумування, яка додає усі значення у полі. Але, за необхідності, тип обчислень, запропонований Microsoft Excel можна змінити.

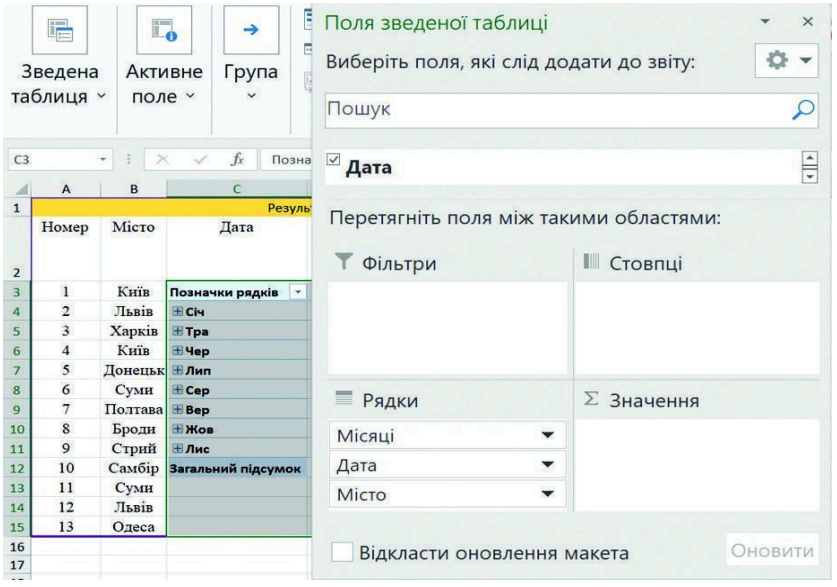


Рис. 14.3. Макет зведеної таблиці і список полів зведеної таблиці

Для обчислення значень у звіті зведеної таблиці, можна використовувати такі типи методів:

– функції зведення у полях значень; – налаштовуванні обчислення; – формули.

Функції зведення у полях значень. Дані в області значень підсумовують вихідні дані у звіті зведеної таблиці.

1. Для вставлення функції зведення необхідно у списку полів зведеної таблиці у розділі **Значення** натиснути кнопку (вставити) відповідного поля та виконати команду **Параметри значення поля**.
2. У однойменному вікні, що відкриється (рис. 14.4), на вкладці **Підсумування значення** за вибрати функцію для розрахунку значень: **Сума, Кількість, Середнє, Максимум** тощо.

Налаштовуванні обчислення. Налаштовувано обчислення показує значення залежно від інших елементів або комірок в області даних.

1. Для вставлення функції таких обчислень необхідно у списку полів зведеної таблиці в розділі **Значення** натиснути кнопку відповідного поля та виконати команду **Параметри значення полів**.
2. У вікні, що відкриється, на вкладці **Відображати значення** як вибрати одну із функцій: **частка загальної суми, частка суми рядків,**

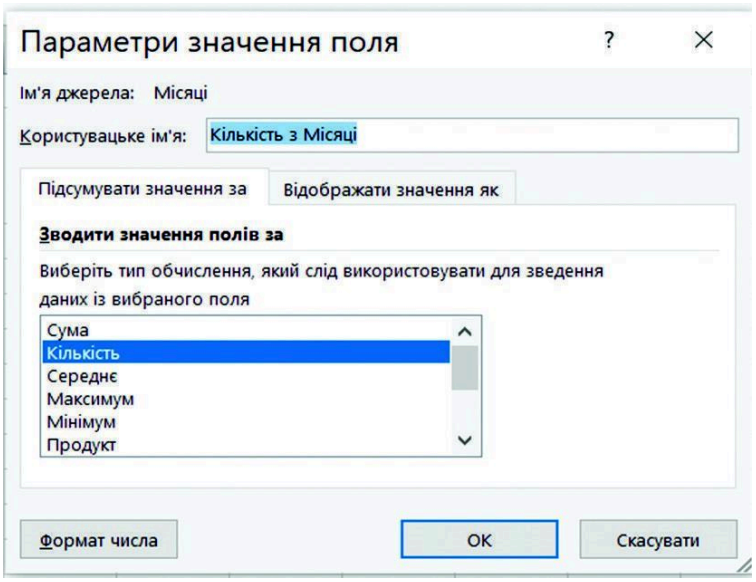


Рис. 14.4. Вікно Параметри значення поля

зі зростаючим підсумком у, сортувати від найменшого до найбільшого, тощо. Microsoft Excel зробить перерахунок значень зведеної таблиці.

Формули. Якщо функції зведення та налаштовуванні обчислення не надають потрібних результатів, можна створити власні формули у полях за якими проводяться обчислення.

Для цього потрібно:

1. Виділити будь-яку комірку всередині зведеної таблиці.

2. На вкладці стрічки **Аналіз зведених таблиць** у групі **Обчислення** натиснути кнопку **Поля, елементи та набори** і виконати команду **Обчислювальне поле**.

3. У вікні **Вставлення обчислювального поля** у полі **Ім'я** ввести ім'я елемента, який буде створюватися, а у полі **Формула** сформувати вираз, на основі якого буде проводитися обчислення (формула починається зі знака «=») (рис. 14.5). Натиснути кнопку **Додати**. Microsoft Excel автоматично додасть обчислюване поле у список полів зведеної таблиці і вставить його у розділ **Значення**. В результаті створене поле з'явиться у зведеній таблиці.

Інші операції в зведених таблицях.

Оновлення зведеної таблиці. Зведена таблиця динамічне пов'язана з базою даних, яку було використано під час її створення. Якщо значення у базі даних змінилися, треба виконати команду: **Дані** ⇨ **Оновити все**. Проте цей спосіб не застосовується, якщо у базі даних з'являються нові рядки або стовпчики. У цьому разі необхідно повернутися до майстра зведених таблиць і зазначити новий діапазон записів, що додаються до таблиці.

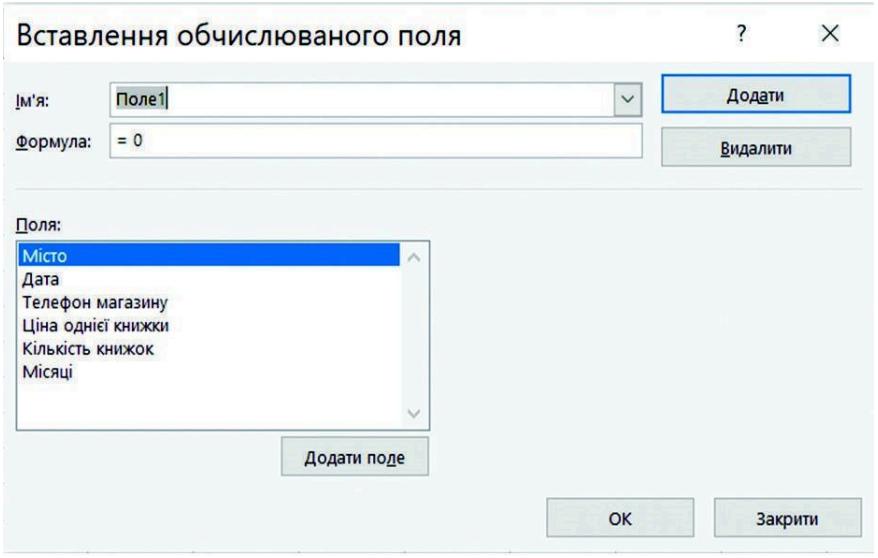


Рис. 14.5. Вікно Вставлення обчислювального поля

Перейменування полів та елементів. Імена полів та елем ентів можна змінити, відредагувавши їх заголовки у зведеній таблиці.

Сортування елементів. Майстер зведених таблиць автоматично упорядковує елементи за їх зростанням. Якщо потрібно впорядкувати елементи за спаданням або за значеннями поля даних, необхідно скористатись командою **Дані** ⇨ **Сортувати**.

Групування елементів. Microsoft Excel автоматично групує елементи внутрішнього поля для кожного заголовка зовнішнього поля і у разі потреби створює проміжні підсумки для кожної групи елементів внутрішнього поля. Проте, іноді виникає потреба групувати елементи в інший спосіб. Щоб створити групу, слід виділити її елементи і виконати команду **Дані** ⇨ **Структура** ⇨ **Групувати**.

Закріплення областей аркуша

Якщо уся таблиця не розміщується на екрані, а потрібно, щоб певна область аркуша залишалась видимою під час його прокручування можна закріпити необхідні рядки та стовпчики, з метою блокування їх на місці або поділу аркуша на області.

Закріплення рядків та стовпців. Закріпити можна лише рядки чи стовпчики, які знаходяться відповідно зверху та зліва на аркуші. Неможливо закріпити рядки чи стовпчики, що знаходяться у середині аркуша. Можна закріпити тільки верхній рядок аркуша, тільки лівий стовпчик або ж декілька рядків чи стовпчиків одночасно. Для цього необхідно виконати одну з наступних дій:

- для закріплення лише одного рядка, виконують команду:

Подання – Закріпити області – Закріпити верхній рядок (рис. 14.6);

– для закріплення лише одного стовпчика, виконують команду:

Подання Закріпити області Закріпити перший стовпець;

- для закріплення декількох рядків та стовпців одночасно, виділяють комірку, вище якої будуть розташовані закріплені рядки і лівіше якої будуть розташовані закріплені стовпчики, та виконують команду:

Подання ⇨ Закріпити області ⇨ Закріпити області;

- для закріплення декількох рядків (починаючи з першого), виділяють рядок під останнім рядком, який потрібно закріпити та виконують команду: **Подання Закріпити області Закріпити області**;
- для закріплення декількох стовпчиків, виділяють стовпчик справа від останнього стовпчика, який потрібно зафіксувати, та виконують команду: **Подання Закріпити області Закріпити області**.

Межа сітки під закріпленими рядками і справа від закріплених стовпчиків буде потовщеною.

Зняти закріплення можна командою **Подання Закріпити області Звільнити області**.

Розділення аркуша на області. Розділення областей – варіант закріплення областей, під час якого, за умови поділу вікна Microsoft Excel створюються дві або чотири окремі області аркуша, які можна окремо прокручувати.

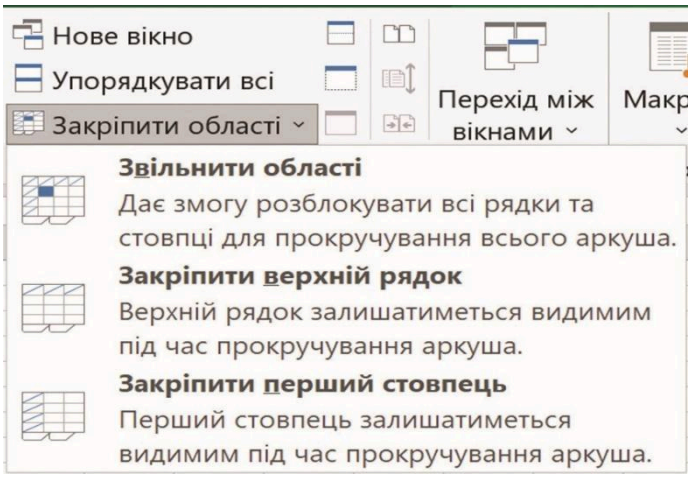


Рис. 14.6. Закріплення рядків та стовпців

Контрольні питання

1 За якими правилами вводяться дані у список електронної таблиці?
2 Як здійснюється сортування списків?
3 Як здійснюється фільтрування даних за допомогою Автофільтра?
4 . Як здійснюється фільтрування даних із застосуванням розширеного
5 фільтра?
6 Як здійснюється автоматичне підбиття підсумків?
7 Як створити діаграму на основі даних, що містяться на робочому
8 аркуші?
9 Які існують основні типи діаграм?
10 З яких елементів складаються діаграми?
11 Як на діаграму додати лінію тренду?
12 Які типи ліній тренду існують?
13 Як здійснюється форматування діаграми?

Лекція 15. Візуалізація рішення економічних завдань за допомогою дашбордів.

15.1 Створення діаграм на основі робочій книги

Ділова графіка – це спеціальний клас графічних зображень, що дають змогу представляти у наочній формі числові дані та оздоблювати своєрідними графічними коментарями зображення різного призначення. Найчастіше ділова графіка використовується під час підготовки різноманітних звітів, доповідей, презентацій, статистичних зведень тощо.

Доволі часто об’єкти ділової графіки використовуються під час підготовки наукової та навчальної літератури.

Створення діаграм. *Діаграми* – засіб наглядного представлення інформації, заданої у вигляді чисел. Діаграми створюються на основі чисел, що містяться у робочому аркуші. Одна діаграма може використовувати дані з будь-якої кількості аркушів і навіть з будь-якої кількості робочих книг. Діаграма зберігає зв’язок з даними, на основі яких вона побудована, і під час зміни цих даних відразу змінюється і вигляд діаграми.

Табличні процесори генерують діаграми автоматично, а користувачу лишається тільки вибрати діапазон комірок, вказати тип діаграми та оформити отриманий графічний об’єкт за власним смаком.

Для того, щоб створити діаграму в Microsoft Excel, необхідно:

1. Увести числові дані на аркуш одним із наступних способів:
   - безпосередньо в програмі Microsoft Excel;
   - скопіювати через буфер обміну;
   - імпортувати з документів, що створені у інших програмах (команда **Дані** ⇨ **Отримання зовнішніх даних**).
2. Виділити діапазон даних, на основі яких буде будуватися діаграма.

Під час виділення у діапазон потрібно включати заголовки рядків і стовпчиків, які будуть відображені як підписи рядів даних діаграми.

Дані для діаграми не обов’язково мають бути розміщені в одному діапазоні. Щоб виділити декілька діапазонів, потрібно натиснути клавішу [CTRL] і вибрати потрібні діапазони комірок. Якщо вибрати лише одну комірку, Microsoft Excel автоматично побудує діаграму на основі суміжних з нею комірок, які містять дані.

3. Вибрати потрібний тип діаграми, який визначається розташуванням даних у комірках таблиці.

Для цього необхідно на вкладці **Вставлення** у групі **Діаграми** вибрати тип та підтип діаграми (рис. 15.1) або натиснути кнопку та у діалоговому вікні **Вставлення Діаграми** на вкладці **Усі діаграми** (рис. 15.2) вибрати потрібний тип та підтип діаграми.

Рис.15.1. Вибір типу діаграми

Для створення організаційної діаграми необхідно на вкладці **Вставлення** натиснути кнопку **SmartArt** та у діалоговому вікні **Вставлення Діаграми** вибрати тип об’єкта.

Для швидкого створення діаграми можна скористатися *за допомогою команд стрічки* комбінацією клавіш [ALT]+[F1].

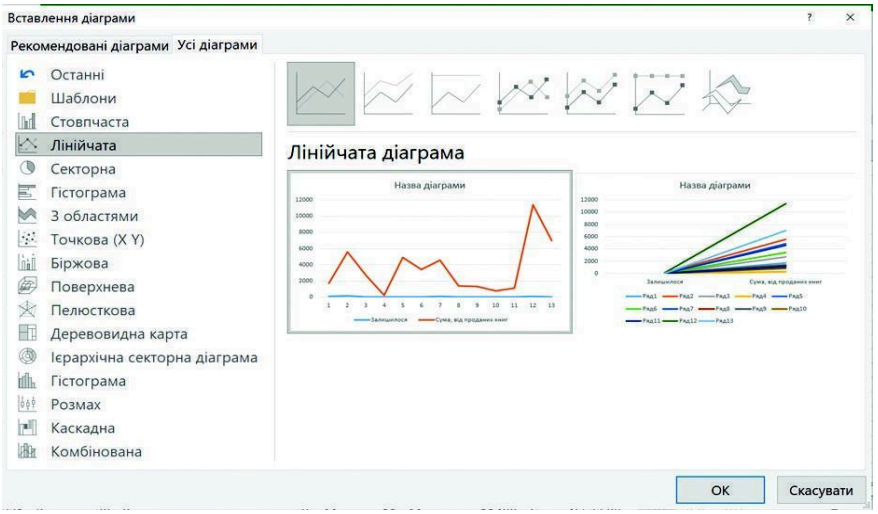


Рис. 15.2. Вибір типу діаграми за допомогою діалогового вікна Вставлення діаграми

Розміщення діаграм. Під час створення діаграми в Microsoft Excel існує два варіанти їх розміщення:

– вставити діаграму безпосередньо на аркуш як один з його об’єктів – *вбудована діаграма* (виконується за замовчуванням); – створити діаграму на новому аркуші робочої книги.

Для зміни варіанту розміщення діаграми необхідно активізувати діаграму та на вкладці **Конструктор Діаграм** у групі **Розташування** натиснути кнопку **Перемістити діаграму**. Відкриється діалогове вікно, у якому вибирається потрібний варіант розміщення.

Діаграма, що розміщена на окремому аркуші, займає весь цей аркуш. Тому, якщо планується друкувати лише одну діаграму на сторінці, найкраще використати цей спосіб. Якщо необхідно створити багато діаграм, тоді можна будувати кожну з них на окремому аркуші. Крім того, цей спосіб дає змогу доволі легко розшукувати потрібну діаграму, надаючи аркушам діаграм відповідні назви.

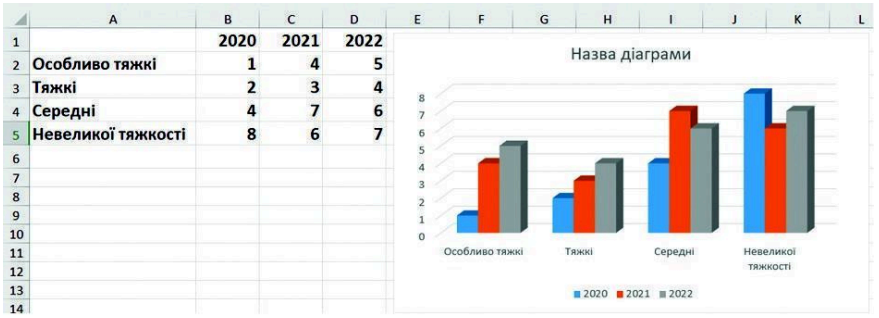
Типи діаграм

Гістограма – спосіб графічного представлення табличних дискретних даних та наочного порівняння різних величин.

На гістограмах категорії зазвичай відкладаються по горизонтальній осі, а значення – по вертикальній.

– *Гістограма з групуванням* – порівнює значення за категоріями та виводить їх у вигляді вертикальних прямокутників (паралелепіпедів, циліндрів, конусів, пірамід).

Формат даних. Кожний рядок в таблиці відповідає окремому стовпчику на діаграмі. У перший стовпчик таблиці вводяться категорії усіх рядків. Решта стовпчиків мають містити числові дані. Для кожного стовпчика даних можуть вказазуватися назви, які автоматично виводяться на діаграмі як підписи осей (рис. 15.3).



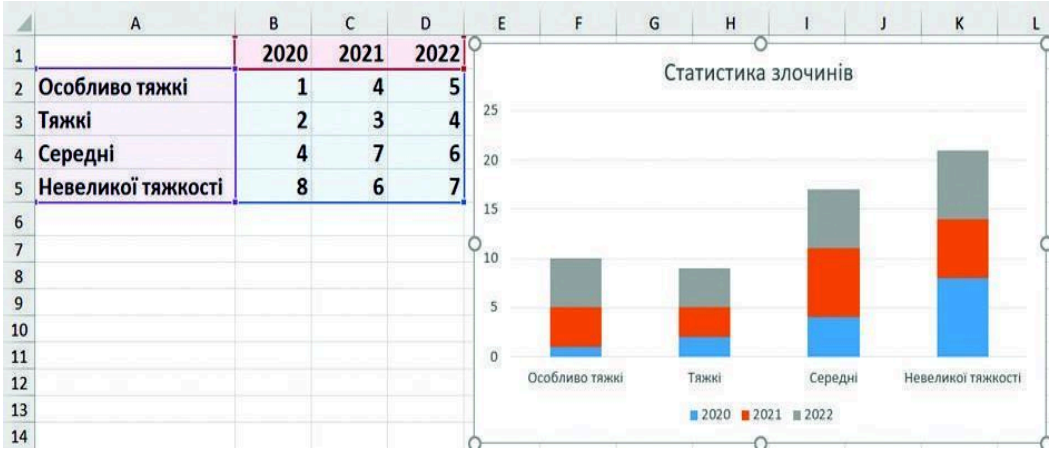
	2020	2021	2022
Особливо тяжкі	1	4	5
Тяжкі	2	3	4
Середні	4	7	6
Невеликої тяжкості	8	6	7

Рис. 15.3. Гістограма з групуванням

– *Гістограма з накопиченням* – демонструє відношення окремих складників до їх сукупного значення, порівнюючи за категоріями внесок кожної величини у загальну суму, та виводить їх у вигляді вертикальних прямокутників з накопиченням.

Такою гістограмою користуються, якщо є декілька рядів даних і потрібно зробити наголос на загальному підсумку.

Формат даних. Дані цього типу діаграм повинні мати ту ж структуру, що і гістограми з групуванням. Однак кожна категорія відповідає лише одному стовпчику, який містить усі дані для визначення категорії і графічно відображає відношення між частинами і цілим значенням (рис. 15.4).



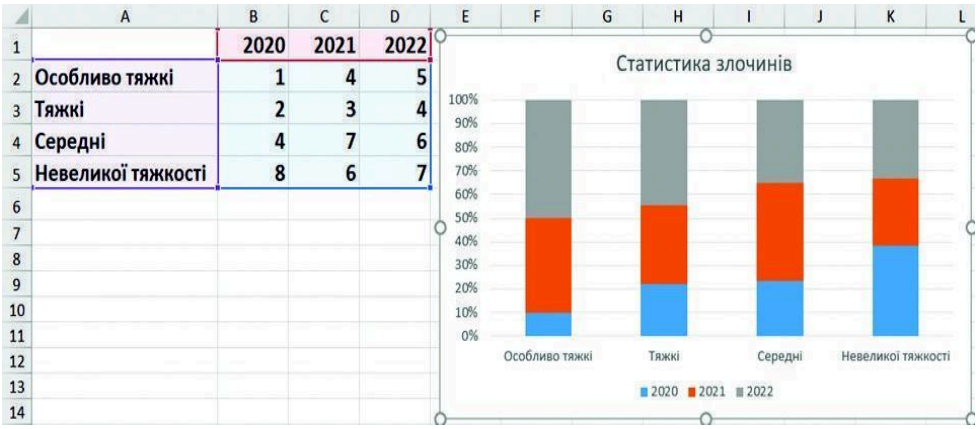
	2020	2021	2022
Особливо тяжкі	1	4	5
Тяжкі	2	3	4
Середні	4	7	6
Невеликої тяжкості	8	6	7

Рис. 15.4. Гістограма з накопиченням

– *Нормована гістограма з накопиченням* – порівнює за категоріями відсотковий вклад кожної величини у загальну суму та виводить їх у вигляді вертикальних прямокутників (паралелепіпедів, циліндрів, конусів, пірамід) з накопиченням.

До такої гістограми звертаються, якщо є три або більше рядів даних і потрібно зробити наголос на їх внеску до цілого, особливо якщо підсумок однаковий для кожної категорії.

Формат даних. Дані цього типу діаграм повинні мати ту ж структуру, що і гістограми з накопиченням. Однак кожна категорія відповідає двом і більше стовпчикам та підсумкові значення однакові для кожної категорії (рис. 15.5).

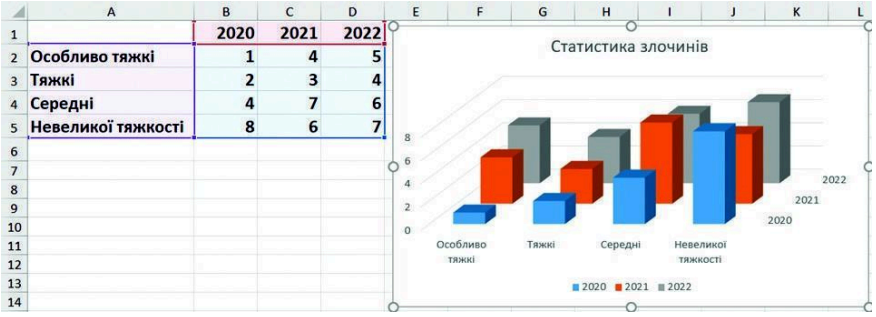


	2020	2021	2022
Особливо тяжкі	1	4	5
Тяжкі	2	3	4
Середні	4	7	6
Невеликої тяжкості	8	6	7

Рис. 15.5. Нормована гістограма з накопиченням

– *Об'ємна гістограма* – використовується для порівняння даних і за категоріями, і за рядками. Вона містить три осі, які можна змінювати (горизонтальна, вертикальна та вісь глибини).

Формат даних. Кожен рядок у таблиці відповідає декільком стовпчикам на діаграмі, а кожен стовпчик декільком рядкам. У перший стовпчик таблиці вводяться категорії усіх рядків. У перший рядок таблиці вводяться категорії усіх стовпчиків. Для кожного стовпчика і рядка даних можуть вказуватися назви, які автоматично виводяться на діаграмі як підписи осей (рис. 15.6).



	2020	2021	2022
Особливо тяжкі	1	4	5
Тяжкі	2	3	4
Середні	4	7	6
Невеликої тяжкості	8	6	7

Рис.15.6. Об'ємна гістограма

Лінійна діаграма. *Лінійна діаграма* – це гістограма, яку повернуто на 90 вправо. Використовується у разі, коли мітки осей та значення, що виводяться, мають велику довжину.

Графіки. *Графіки* – тип діаграм, що застосовуються для відображення неперервних даних в єдиному масштабі. *Графіки з маркерами* – тип діаграм, що застосовуються для відображення неперервних даних з відміченими окремими значеннями.

Графіки зручно використовувати у разі, коли у нижній частині діаграми потрібно відобразити дати для демонстрації хронології розвитку події, а також під час порівняння змін значень у декількох категоріях даних.

Формат даних. У перший стовпчик вводять назви категорій, що представлені на графіку. Решта стовпчиків мають містити числові дані, які відповідають лініям діаграми. На графіку може відображатися один або декілька таких стовпчиків. Для кожного стовпчика можна вказати назву (здебільшого місяць, рік, квартал, дата), яка автоматично з'являється в якості підписів даних на діаграмі. На графіках дані категорій рівномірно розподіляються по горизонтальній осі, а всі значення рівномірно розподіляються вздовж вертикальної осі.

Так само як і для гістограм, розрізняють: графік, графік з накопиченням, нормований графік з накопиченням та об'ємний графік (рис. 15.7).

Рис. 15.7. Графіки

Діаграми з областями. Діаграма з областями підкреслює величину змін з часом та використовуються для акцентування на сумарному значенні. Подібна на розмальований різними кольорами графік.

Кругові діаграми. *Кругова діаграма* – тип діаграм, що дозволяє графічно представляти дані як сегменти кола або відсоткові частини від цілого значення. Вона демонструє розмір елементів одного ряду даних щодо суми усіх елементів.

Така діаграма не використовується під час зіставлення величин, які немає сенсу сумувати (наприклад, середні значення). У таких випадках використовують або гістограму (для порівняння окремих величин, або графік (для зіставлення змін величин за деякий проміжок часу).

Формат даних. Для усіх підтипів кругової діаграми дані мають міститися у двох стовпчиках або рядках таблиці. У першому вказується категорія, а у другому – відповідне числове знач ення (рис. 15.8).

Рис. 15.8. Кругова діаграма

Розрізняють: кругову, вторинну кругову (для перегляду невеликих секторів головної кругової діаграми) та розрізану кругову діаграму. Може бути представлена в дво - або тривимірному вигляді.

Кільцеві діаграми. Кільцеві діаграми нагадують кругові діаграми з вирізаною серединою. Однак важлива різниця є в тому, що такі діаграми можуть представляти декілька наборів даних. Сприймати інформацію за допомогою кільцевої діаграми достатньо складно, тому рекомендується замість них використовувати гістограми з накопиченням або лінійні діаграми з накопиченням.

На діаграмах цього типу дані відображаються у вигляді кілець, кожне з яких представляє ряд даних. Якщо дані представляються у відсотках, то кожне кільце у сумі повинно давати 100%.

Формат даних. Для всіх підтипів кільцевої діаграми дані повинні міститися у більше як в двох стовпчиках або рядках таблиці. У першому вказується категорія, а в інших – відповідні числові значення (рис. 15.9).

Точкові діаграми. Точкова діаграма – відображає відношення між числовими значеннями у кількох рядах даних: один набір числових даних уздовж горизонтальної осі (x), а інший – уздовж вертикальної осі (y). Ці значення поєднуються в єдині точки й відображаються в нерівних інтервалах або групах. Точкові діаграми зазвичай застосовуються для відображення та порівняння наукових, статистичних або інженерних даних.

Розрізняють: точкові діаграми з маркерами, точкові діаграми із плавними лініями, точкові діаграми з прямими лініями.

	2014	2015	2016
Особливо тяжкі	5	7	10
Тяжкі	20	15	5
Середньої тяжкості	50	100	50
Невеликої тяжкості	125	90	75

Рис. 15.9. Кільцеві діаграми

Бульбашкові діаграми. Бульбашкова діаграма, на відміну від інших діаграм, використовується для порівняння трьох наборів значень. Такий додатковий набір даних відображається у вигляді розміру бульбашки.

Формат даних. Дані розташовуються у трьох стовпчиках. У першому стовпчику – значення X, у другому стовпчику – відповідне значення Y і в третьому стовпчику значення, що визначають розмір бульбашки.

Біржові діаграми. Біржові діаграми використовуються найчастіше у банківській справі для відображення коливань курсу валют, вартості акцій на біржі тощо. Однак їх можна використовувати і для виводу наукових чи статистичних даних.

Формат даних. У перший стовпчик таблиці вводяться значення дат, а у наступні три – набори даних у певному порядку: наприклад (рис. 15.10), найвищий курс, найнижчий курс, курс закриття.

Дата	Найвищий курс	Найнижчий курс	Курс закриття
1 січ.	17,20	13,49	15,75
2 січ.	15,03	9,05	13,05
3 січ.	13,97	14,50	13,00
4 січ.	12,01	8,20	8,80

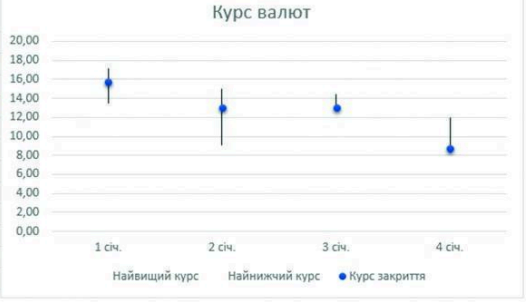


Рис. 15.10. Біржова діаграма

Поверхневі діаграми. Поверхнева діаграма призначена для знаходження оптимальної комбінації даних з двох наборів. На відміну від діаграм інших типів, у них колір використовується для виділення значень, а не різних наборів даних. Області, що належать до однакових діапазонів виділяються однаковими кольорами та візерунками.

Розрізняють: дротова (без кольору на поверхні), об’ємна (демонструють зміну значень даних за двома вимірами у вигляді поверхонь), контурна (вигляд зверху).

Пелюсткові діаграми. Пелюсткова діаграма дає змогу порівнювати значення декількох рядів даних. Водночас категорії відображаються на окремих променях двомірного графіку. Точки даних з таблиці відкладаються на відповідних променях і з’єднуються лініями. Усі промені виходять з центру діаграми.

Формат даних. Дані організовуються у вигляді одного або декількох стовпчиків. Перший стовпчик є необов’язковим. У нього вводяться текстові дані для позначення променів. Решта стовпчики мають містити числові дані, які будуть відкладатися уздовж променів діаграми.

Комбіновані діаграми. Комбінована діаграма використовується для суміщення в одній діаграмі різних типів діаграм.

Отже, Microsoft Excel підтримує різні типи діаграм, що дає можливість представляти дані найбільш зрозумілим для тієї чи іншої аудиторії способом.

Редагування діаграми. Після створення діаграми можна змінити будь-який з її елементів. Наприклад, можна змінити спосіб відображення осей, додати назву діаграми, перемістити або приховати легенду, відобразити додаткові елементи діаграми тощо.

Для проведення цих операцій необхідно активізувати діаграму та на вкладці **Конструктор Діаграм** натиснути кнопку **Додати елемент діаграми** та вибрати відповідну команду з меню, що відкриється (рис. 15.11).

Для швидкого видалення елемента діаграми його необхідно виділити та натиснути клавішу **Delete**.

Для зв’язування назви діаграми або осі з коміркою аркуша необхідно виділити назву, яку потрібно прив’язати до комірки, розмістити вказівник у рядок формул аркуша, ввести знак рівності (=) та вибрати комірку аркуша з даними або текстом, який потрібно відобразити у назві. Натиснути клавішу **Enter**.

Форматування діаграми. Щоб надати діаграмі нестандартного вигляду та привернути до неї увагу, можна застосувати форматування до окремих елементів діаграми, наприклад, до маркерів даних, області діаграми, області побудови, чисел, тексту назв і підписів. Усі ці операції зібрані на вкладці стрічки **Формат** і дають змогу здійснити такі операції:

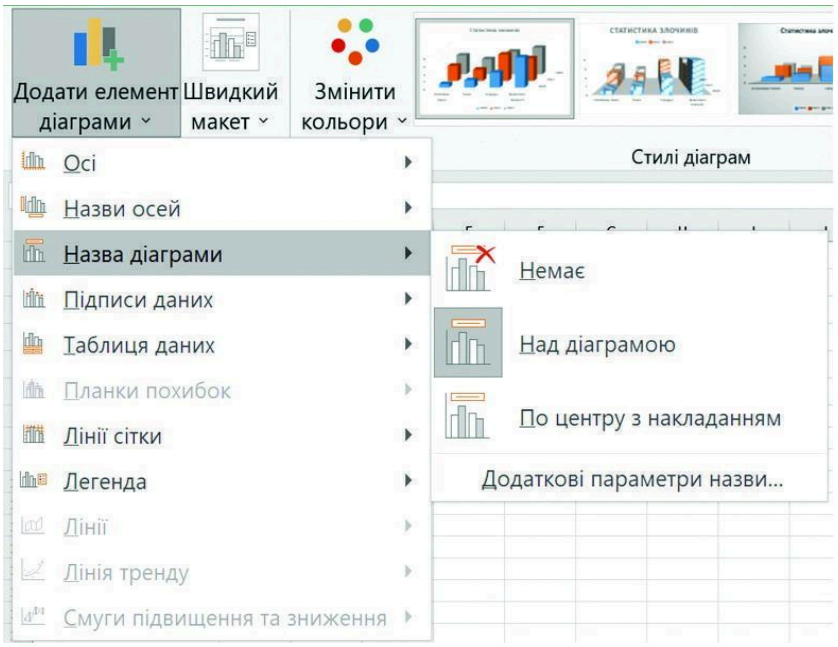


Рис. 15.11. Меню редагування діаграми

– *заливка елементів діаграми*. Щоб зосередження увагу до певних елементів діаграми, можна використовувати кольори, текстури, рисунки та градієнтну заливку;

– *змінення контурів елементів діаграми*. Для акцентування на окремих елементах діаграми, можна змінювати кольори, стилі та товщину ліній;

– *додавання додаткових ефектів до елементів діаграми*. Щоб надати діаграмі завершений вигляд, до фігур діаграми можна застосувати додаткові ефекти – тіні, відбиття, світіння, згладжування, рельєф, ефект об'ємного обертання;

– *форматування тексту і чисел*. Можна форматувати текст і числа в назвах, підписах і текстових полях на діаграмі так само, як текст і числа на аркуші. Щоб зробити текст і числа виразнішими, можна застосувати стилі **WordArt**;

– *переміщення або змінення розміру діаграми*. Діаграму можна переміщати у будь-яке місце на аркуші методом перетягування або ж на інший аркуш. Також за потреби можна змінювати розмір діаграми (перетягнути маркери змінення розміру або на вкладці **Формат** ввести потрібні розміри в поля групи **Розмір**).

Автоматичне редагування та форматування діаграм. Щоб не редагувати чи не форматувати діаграму у ручному режимі, користуються попередньо визначеними макетами та стилями, які вже містяться у програмі Microsoft Excel. За потреби макет або стиль можна налаштування самостійно, змінюючи окремі елементи діаграми (наприклад, область діаграми чи область побудови).

У разі застосування попередньо визначеного макета, у діаграмі певний набір елементів впорядкується деяким способом. Для кожного типу діаграми є багато різних макетів.

Microsoft Excel пропонує різноманітні корисні колекції макетів (вкладка **Конструктор діаграм** ⇨ група **Макети діаграм**) і стилів (вкладка **Конструктор діаграм** група **Стилі діаграм**) (або експрес-макети та експрес-стилі), з яких не лише можна вибрати потрібні варіанти, але й змінити їх, відредагувавши макет або формат окремих елементів діаграми.

Самостійно створювати макети та стилі діаграм неможливо, проте можна створити шаблон діаграми та вкласти до нього макет і потрібне форматування.

15.2 Аналіз даних за допомогою діаграм

Для різних типів діаграм доступні додаткові лінії (наприклад, коридор коливань і лінії тренду), смуги (наприклад, смуги підвищення/зниження та планки похибок), позначки даних та інші параметри.

Оброблення пропущених даних. Інколи у наборі даних для діаграми бувають пропущені дані. В Microsoft Excel передбачено декілька операцій для виправлення такої ситуації.

Для початку необхідно активізувати діаграму та виконати послідовність команд: **Конструктор діаграм** ⇨ **Вибрати дані** ⇨ **Приховані та пусті клітинки**.

У вікні **Настройки прихованих і пустих клітинок** (рис. 15.12), що відкриється, вибрати один із параметрів:

- *пробіли* – пропущені дані просто ігноруються і набір даних буде мати проміжок;
- *нуль* – пропущені дані сприймаються як нуль;
- *з'єднувати точки даних лінією* (значення інтерполюються) – пропущені дані обчислюються на основі вибірок. Ця опція доступна лише для графіків;
- *відображати дані у прихованих рядках і стовпцях* – у діаграмі відображаються або приховуються приховані рядки під час роботи із структурою або відфільтрованими даними.

Додавання лінії тренду. Під час побудови діаграм для даних, що залежить від часу, можна будувати лінію тренду, яка буде відображати тенденцію розвитку даних. У деяких випадках за допомогою лінії тренду можна прогнозувати зміни даних. Окремі набори даних можуть мати декілька ліній тренду.

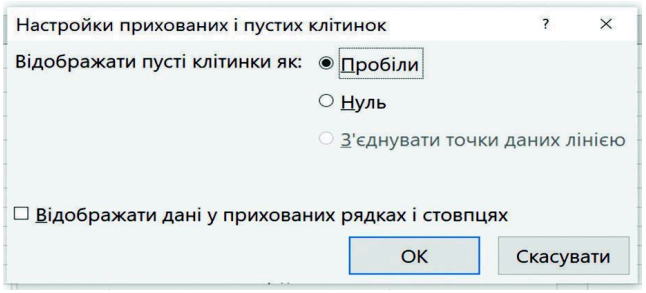


Рис. 15.12. Вікно Настройки прихованих і пустих клітинок

Для встановлення лінії тренду необхідно активізувати діаграму та виконати послідовність команд **Конструктор діаграм** ⇨ **Додати елемент діаграми** ⇨ **Лінія тренду** (див. рис. 15.13)

Є такі типи лінії тренда: лінійна, експоненціальна, логарифмічна, поліноміальна, степенева та лінійна фільтрація. Тип лінії тренду, який слід вибирати, визначається типом наявних даних.

Лінія тренду отримується найточнішою, коли її величина вірогідності апроксимації (наближення) наближується до одиниці. Під час апроксимації даних за допомогою лінії тренду величина вірогідності апроксимації розраховується Microsoft Excel автоматично. За необхідності отриманий результат можна відобразити на діаграмі. Для цього необхідно викликати контекстне меню лінії тренду, виконати команду **Формат лінії тренду** та встановити прапорець **Розмістити на діаграмі величину апроксимації**.

Лінійна лінія тренду – це пряма лінія, яка наближено описує сукупність даних. Вона використовується у найпростіших випадках, коли дані розташовані близько до прямої. Інакше кажучи застосування лінійної апроксимації зручно використовувати для величини, яка збільшується або зменшується з постійною швидкістю. У наведеному далі прикладі (рис. 15.12) пряма лінія тренду описує стабільний ріст правопорушень протягом трьох років. Величина достовірності апроксимації дорівнює 0,9643, що свідчить про хороше зіставлення розрахованої лінії з даними.

Логарифмічна лінія тренду – добре описує величину, яка спочатку швидко зростає або спадає, а далі поступово стабілізується. Наприклад, прогноз росту, зниження правопорушень у певному регіоні.

Поліноміальна лінія тренду – використовується для опису величин, почергово зростаючих та спадаючих. Вона корисна, на приклад, для аналізу великого набору даних про нестабільну величину. Наприклад, залежність витрати палива від швидкості руху (рис. 15.13).

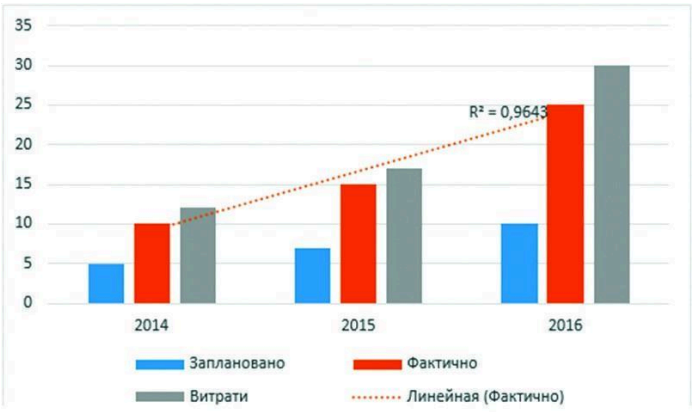


Рис.15.12. Лінійна лінія тренду

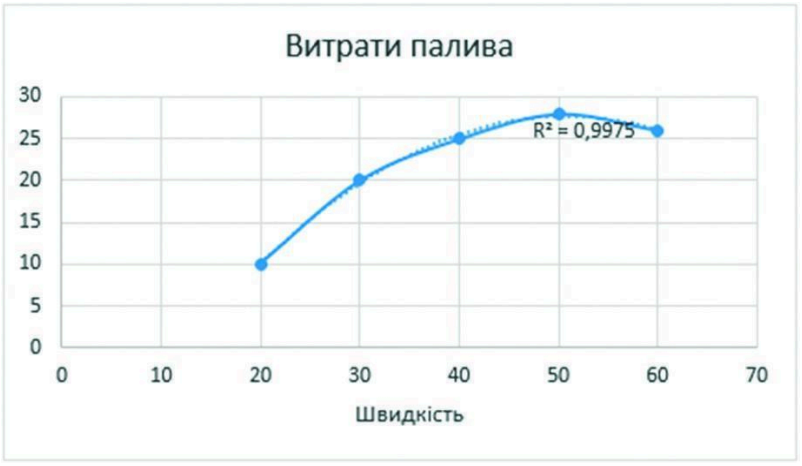


Рис. 15.13. Поліноміальна лінія тренду

Степенева апроксимація – корисна для опису монотонно зростаючої або монотонно спадної величини, наприклад відстань, пройдена автомобілем, що прискорюється.

Експоненціальна лінія тренду – використовується у тому разі, якщо швидкість зміни даних постійно зростає. Наприклад, розрахунок складних відсотків у аналізі даних.

Лінійне фільтрування дає змогу згладити коливання даних і так наочніше показати характер залежності. ЇЇ застосовують, коли точок є багато і вони дуже розкидані. Дані усереднюються попарно чи в інший спосіб і одержані середні значення використовуються як точки лінії тренду (рис. 15.14).

Лінійний прогноз – дозволяє за допомогою лінії тренду графічно продемонструвати тенденцію зміни даних. Наприклад, якщо в Microsoft Excel є діаграма, що відображає певні дані за перші декілька років, можна додати до цієї діаграми лінію тренду, яка відображатиме загальні зміни цих даних (зростаючу, спадну або рівну) або очікувані зміни в наступних роках (рис. 15.15).

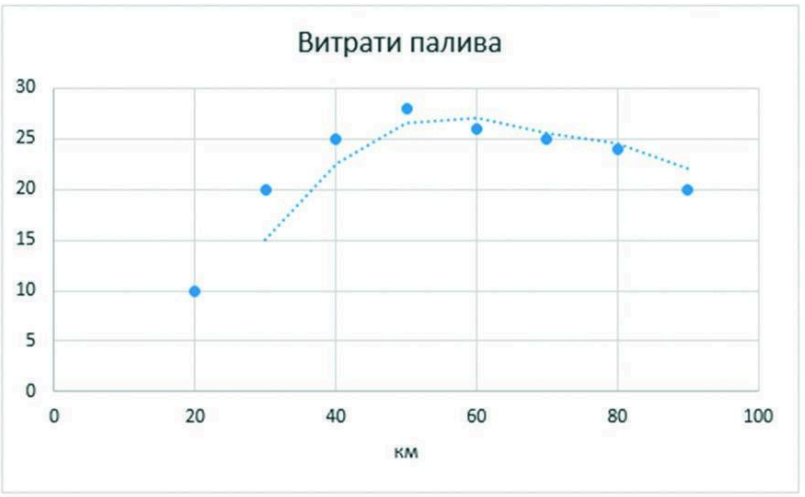


Рис. 15.14. Лінійне фільтрування

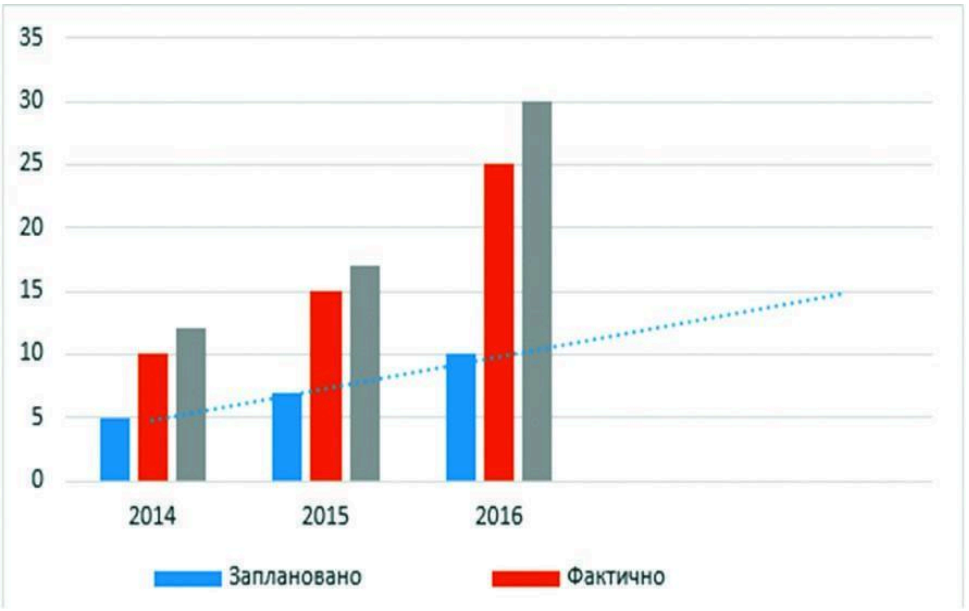


Рис. 15.15. Лінійний прогноз

Контрольні питання

1. З яких елементів складається графічний інтерфейс програми Microsoft Excel?
2. Які основні типи даних користувач може ввести у комірки робочої книги Microsoft Excel?
3. Якими способами можна виділити комірки робочої книги Microsoft Excel?
4. Як створити діаграму на основі даних, що містяться на робочому
5. аркуші?
6. Які існують основні типи діаграм?
7. З яких елементів складаються діаграми?
8. Як здійснюється оброблення пропущених даних на діаграмі?
9. Як на діаграму додати лінію тренду?
10. Які типи ліній тренду існують?
11. Як здійснюється форматування діаграми?

СПИСОК ЛІТЕРАТУРИ

1. Веб-портал. Здобуваємо знання разом. https://webportal.com.ua/category/microsoft-office/excel/
2. ІНФОРМАТИКА. Навчально-методичні матеріали для студентів 1 курсу. https://surl.li/kaamlw
3. Магеровська Т. В. Електронні таблиці та системи управління базами даних: навчальний посібник / Т. В. Магеровська, Я. М. Пелех, В. В. Сеник, А. В. Кунинець. Львів : Самвидав, 2020. 415 с.
4. Інформаційні технології : навчальний посібник / О. І. Зачек, В. В. Сеник, Т. В. Магеровська та ін.; за ред. О. І. Зачека. Львів: Львівський державний університет внутрішніх справ, 2022. 432 с.
5. Конспект лекцій з дисципліни «Прикладні соціокомунікаційні технології» [для здобувачів денної, заочної та дистанційної форм навчання спеціальності 029 Інформаційна, бібліотечна та архівна справа] / Укл.: С.М. Мельник. – Одеса : ОНПУ, 2020. – 111 с.
6. ВЕБ. Розробка та дизайн. Лекції. [Електронний ресурс]. https://www.victoria.lviv.ua/library/students/wd/lecture.html
7. Що краще Віндовс 10 або 11 – порівняння операційних систем. [Електронний ресурс]/ https://nbookpart.com.ua/scho-krasche-vindovs-10-abo-11-porivnyannya-operatsiynyh-system/
8. Операційні системи. Конспект лекцій. [Електронний ресурс]. https://e-tk.lntu.edu.ua/mod/page/view.php?id=3765
9. Історія розвитку текстового процесора Microsoft Word. [Електронний ресурс] https://www.timetoast.com/timelines/2125400
10. Фінансова академія Актів. [Електронний ресурс]. https://finacademy.net/ua/materials/article/istoriia-microsoft-excel
11. Підручники для ВНЗ онлайн. [Електронний ресурс] https://pidru4niki.com/13561021/informatika/informatsiyna_tehnologiya_avtomatizatsiyi_protsesu_analizu_informatsiyi#google_vignette