МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Л. Ю. Маноха
Л.Г. Загоровська
Н.Н. Бровченко

ВИДИ ТА ФОРМИ ПРЕДСТАВЛЕННЯ АЛГОРИТМІВ

КОНСПЕКТ ЛЕКЦІЙ

з дисциплін Алгоритмізація і програмування,
Основи програмування та алгоритмічні мови

для студентів напряму 0925
“Автоматизація та комп’ютерно-інтегровані технології”
та спеціальності 6.080401 “Інформаційні управляючі системи та технології”
денної та заочної форм навчання

Всі цитати, цифровий та фактичний матеріал, бібліографічні відомості перевірені. Написання одиниць відповідає стандартам

СХВАЛЕНО
на засіданні кафедри
інформаційних систем
Протокол № 24
від 23.06.04.

Київ НУХТ 2005

Маноха Л.Ю., Загоровська Л.Г., Бровченко Н.Н. Види та форми представлення алгоритмів: Конспект лекції з дисциплін “Алгоритмізація і програмування” та “Основи програмування та алгоритмічні мови” для студ. напряму 0925 “Автоматизація та комп’ютерно–інтегровані технології” та спец. 6.080401 “Інформаційні управляючі системи та технології” ден. та заоч. Форм навчання. – К.: НУХТ, 2005. – с.

Укладачі: Л. Ю. Маноха,

Л.Г. Загоровська, кандидати техн. наук,

Н.Н. Бровченко

Відповідальний за випуск В. В. Самсонов, канд. техн. наук

ВСТУП

Прийняття рішень на будь-якому рівні автоматизації керування об'єктами чи процесами завжди складається з таких етапів: формування мети; визначення основних властивостей очікуваних результатів; опис вихідної інформації; побудова алгоритму перетворення вихідної інформації у результуючу інформацію із заданими властивостями.

Перші три етапи виконує фахівець відповідної галузі (формує технічне завдання), а четвертий – спеціаліст з автоматизації та побудови комп'ютерної системи.

На сучасному рівні розвитку організації процесів керування практично відсутні окремі задачі обробки інформації. Зазвичай вирішуються комплекси задач, пов'язані між собою єдиною інформаційною базою. Автоматизація технологічного процесу керування нині йде шляхом перерозподілу робіт і функцій між фахівцями відповідної галузі, постановниками задач, проектувальниками систем, програмістами, користувачами, супровідниками програм.

Рівень автоматизації об'єкту керування визначається формальним описом його складових частин, функцій і взаємозв'язків між ними. При цьому значну увагу приділяють опису структур даних, інформаційних потоків, їх функціональним зв'язкам та алгоритмам обробки.

Отже, нині перед фахівцем стоїть завдання не просто навчитися програмувати, а засвоїти принципи організації алгоритмічних процесів і форми їх представлення; засвоїти базові алгоритми проектування процесів пошуку, передавання, обробки, організації та збереження інформації у різних інформаційних технологіях; навчитися проводити аналіз і контроль алгоритмів на різних технологічних стадіях життєвого циклу програми; ознайомитися з основними мовами опису алгоритмів, особливостями програмування в інтегрованому середовищі Turbo Pascal, принципами розроблення та реалізації програм; вміти пояснити результати виконання програми та складати супровідну документацію (звіт).

1. ПОНЯТТЯ АЛГОРИТМУ, ЙОГО ВИЗНАЧЕННЯ ТА ОСНОВНІ ВЛАСТИВОСТІ

Алгоритм є одним з основних понять обчислювальної математики та сучасних інформаційних систем будь-якої галузі. Слід зрозуміти, що поки в уяві або на папері не буде побудована чітка послідовність дій процесу розв'язання будь-якої задачі, не можна описувати цей процес мовою програмування, оскільки зі зростанням складності задачі важко написати відповідну програму.

Під *алгоритмом* розуміють точну та скінчену послідовність дій, що виконуються у певному порядку для розв'язання всіх задач даного класу.

Це означає, що для багатьох задач можна побудувати послідовність їх розв'язання, але, якщо цю послідовність можна використати для розв'язання

ряду подібних задач, то вона наближається до алгоритму. Щоб ця послідовність стала алгоритмом, необхідно ще виконати такі умови:

дискретність, тобто розчленованість алгоритму на окремі операції;

детермінованість – однозначна визначеність результату кожної операції, не залежна від виконавця;

масовість – можливість застосування до будь-яких вихідних даних задач визначеного класу;

результативність – скінченність процесу перетворення інформації.

Отже, щоб побудувати алгоритм, слід дотримуватись таких правил:

вхідні та вихідні дані задаються у вигляді визначеної послідовності символів;

процес розв'язання задачі – це послідовність перетворення вхідних даних у вихідні;

процес перетворення складається із сукупності елементарних допустимих операцій;

елементарна допустима операція – це проста, чисто механічна дія, результат якої не залежить від виконавця (комп'ютера чи людини);

послідовність допустимих операцій визначена однозначно.

2. ФОРМИ ТА ЗАСОБИ ПРЕДСТАВЛЕННЯ АЛГОРИТМІВ

Словесна форма – найстаріша форма представлення (алгоритми Евкліда). Формальні правила перетворення інформації формулюються словами, нумеруються та вказується послідовність їх виконання.

Приклад. Обчислення суми послідовності чисел ряду від 1 до n: S=1+2+..+n.

Розв'язання. 1. Покласти і дорівнює 1 та перейти до п.2.

2. Покласти S дорівнює 0 та перейти до п.3.

3. Збільшити S на і та перейти до п.4.

4. Перевірити, чи і більше за n. Якщо **так**, то обчислення припинити. Якщо **ні**, то збільшити і на 1 та перейти до п.3.

Словесно-формульна форма – використовує загальноприйняті математичні позначення та коментарі, що пояснюють дії.

Приклад. Обчислення суми послідовності чисел ряду від 1 до n: S=1+2+..+n.

Розв'язання. Введемо мітки М1, М2, М3, якими будемо позначати дії. Позначення := означає виконання дії присвоєння значення певній змінній.

S: =0.

i: =0.

М1: S: =S+i.

Якщо I>n, перейти до М3, інакше до М2.

М2: i: =i+1. Перейти до М1.

М3: Закінчити обчислення.

Граф-схема – це представлення алгоритму у вигляді системи точок, кожна з яких визначає дію, та стрілок, що вказують перехід від одної дії до іншої (рис.1).

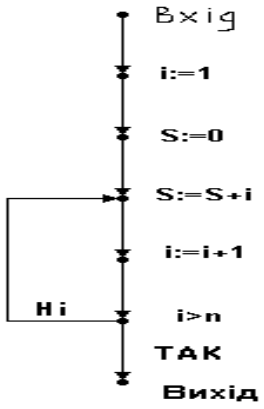


Рис.1. Граф-схема обчислення сумипослідовності чисел ряду від 1 до n

Операторні схема – наближає алгоритм до програмного вигляду – послідовність операторів певних типів.

Приклад. Обчислення суми послідовності чисел ряду від 1 до n: S=1+2+..+n.

Розв'язання. Нехай оператор D обчислює суму S:=S+i, F(i) обчислює i:=i+1. Тоді алгоритм визначення схеми

$$\{0\rightarrow S\}\ \{1\rightarrow i\}\ \downarrow_1 DF(i)p(i>n)\uparrow^1 \text{останов.}$$

HIPO-схема (Hierarchy Input Process Output) – таблиця, кожен рядок якої містить вказівки щодо вхідної інформації, дії над нею та вихідної інформації (рис.2).

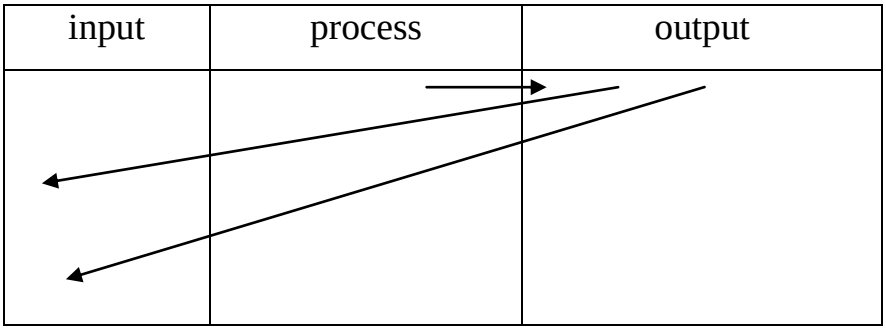


Рис.2. Таблиця для Hipo-схеми

Блок-схема – при використанні цієї форми процес розв'язання поділяється на окремі етапи (або операції), що зображуються у вигляді спеціальних блоків, конфігурація яких вказує на тип дій. Конфігурація та розміщення блоків визначені відповідними стандартами ЄСПД (єдиної системи програмної документації). Блок-схема представлення алгоритму більш поширена, ніж інші форми, бо вона наочна і дозволяє представити алгоритми з різним ступенем деталізації.

3. ЗАГАЛЬНІ ПОЛОЖЕННЯ СКЛАДААННЯ СХЕМ АЛГОРИТМІВ

Схеми алгоритмів, програм, даних і систем (далі – схеми) складаються з символів, що мають задане значення, короткого пояснювального тексту і з'єднувальних ліній.

Схеми можуть використовуватися на різних рівнях деталізації, причому кількість рівнів залежить від розмірів і складності задачі обробки даних. Рівень деталізації повинен бути таким, щоб різні частини і взаємозв'язок між ними були зрозумілі в цілому.

Схема – графічне представлення визначення, аналізу або методу розв'язання задачі, в якому використовуються символи для відображення операцій, даних, потоку, обладнання та ін.

Схеми алгоритмів відображають послідовність операцій у програмі.

Склад схеми такий:

а) символи процесу, що вказують фактичні операції обробки даних, включаючи символи, які визначають шлях, якого треба дотримуватися з урахуванням логічних умов;

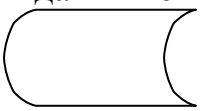
б) лінійні символи, що вказують на потік керування;

в) спеціальні символи, які використовуються для полегшення написання і читання схеми.

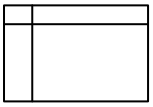
3.1. Основні символи даних

Дані. Символ відображає дані, носій яких не визначено.

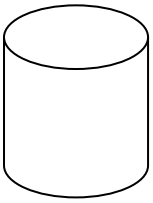
Запам'ятовуючі дані. Символ відображає збережені дані у вигляді, придатному для обробки, носій даних не визначено.



Специфічні символи даних. Оперативно запам'ятовуючий пристрій.Символ відображає дані, які збережені в запам'ятовуючому пристрої.

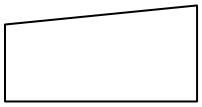


Символ відображає дані, що зберігаються в запам'ятовуючому пристрої **прямого доступу** (магнітний диск, магнітний барабан, гнучкий магнітний диск).



Документ. Символ відображає дані, представлені на носії в зручній для читання формі (машинограма, документ для оптичного або магнітного зчитування, мікрофільм тощо).

Ручне введення. Символ відображає дані, введені вручну за час обробки з пристроїв будь-якого типу (клавіатура, перемикачі, кнопки, світове перо, полоски із штрих-кодом).



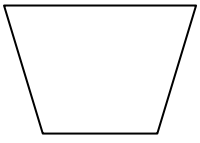
Дисплей. Символ відображає дані, представлені у формі, зручній для читання людиною, на носії у вигляді відображального пристрою (екран для візуального спостереження, індикатори вводу інформації).

3.2.Символи процесу

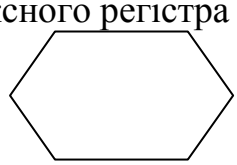
Процес. Символ відображає функцію обробки даних будь-якого виду (виконання певної операції або групи операцій, яке приводить до зміни значення, форми або розміщення інформації або до визначення, за яким з кількох напрямків потоку треба рухатися).

Визначений процес. Символ відображає визначений процес, який складається з однієї або кількох операцій або кроків програми, які визначені в іншому місці (у підпрограмі, модулі).

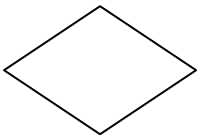
Ручна операція. Символ відображає будь-який процес, виконуваний людиною.



Підготовка. Символ відображає модифікацію команди або групи команд з метою впливу на деяку наступну функцію (установлення перемикача, модифікація індексного регістра або ініціалізація програми).



Рішення. Символ відображає рішення або функцію перемикального типу, яка має один вхід і ряд альтернативних виходів, один і тільки один із яких може бути активований після обчислення умов, визначених всередині цього символу. Відповідні результати обчислення можуть бути записані поруч з лініями, що відображають ці шляхи.



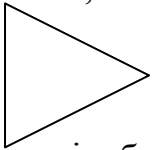
Межа циклу. Символ складається з двох частин, відображає початок і кінець циклу. Обидві частини символу мають один і той самий ідентифікатор. Умови для ініціалізації, прирощення, завершення тощо містяться всередині символу, на початку або в кінці, залежно від розташування операції, що перевіряє умову.

3.3. Символи ліній

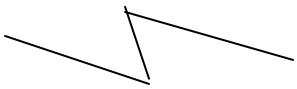
Лінія. Символ відображає потік даних або керування.

У разі потреби або для підвищення наочності можуть бути добавлені стрілки-вказівники.

Передавання керування. Символ відображає безпосереднє передавання керування від одного процесу до іншого, іноді з можливістю прямого повернення до ініціювального процесу після того, як ініційований процес завершить свої функції. Тип передачі керування має бути названий всередині символу (наприклад, запит, виклик, подія).



Канал зв’язку. Символ відображає передавання даних каналом зв’язку.

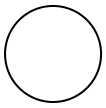


Пунктирна лінія. Символ відображає альтернативний зв'язок між двома або більше символами. Крім того, символ використовують для обведення анотованої ділянки.

– – – – – – – – – – – – – – –

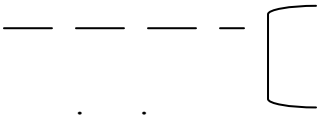
3.4.Спеціальні символи

З'єднувач. Символ відображає вихід у частину схеми і вхід із другої частини цієї схеми і використовується для обриву лінії і продовження її в іншому місці. Відповідні символи-з'єднувачі повинні містити одне і те саме унікальне позначення.



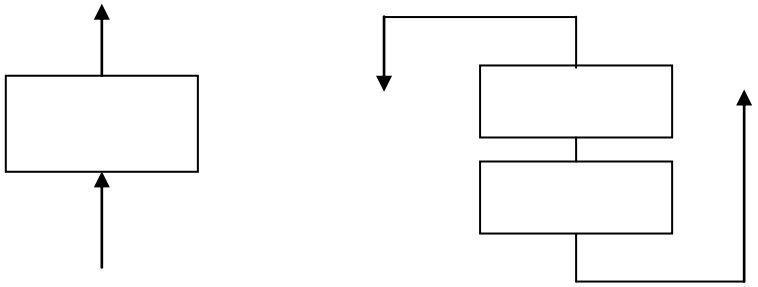
Термінатор. Символ відображає вихід у зовнішнє середовище і вхід із зовнішнього середовища (початок або кінець схеми програми, зовнішнє використання і джерело або пункт призначення даних).

Коментар. Символ використовують для додавання описових коментарів або пояснювальних записів з метою пояснення або приміток. Пунктирні лінії в символі коментаря зв'язані з відповідним символом або можуть обводити групу символів. Текст коментарів або приміток має бути поміщено біля відокремлюючої фігури.



Мінімальна кількість тексту, необхідного для розуміння функції даного символу, слід розмістити всередині даного символу. Текст для читання повинен записуватися зліва направо і зверху вниз незалежно від напрямку потоку.

Приклад.



Якщо обсяг тексту, що розміщується всередині блока, перевищує його розміри, слід використовувати символ коментаря.

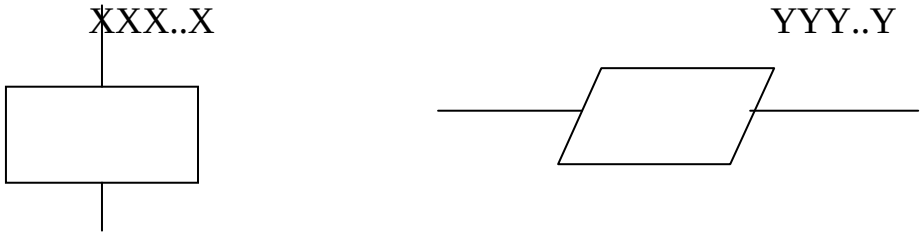
Якщо використання символу коментаря може заплутати або порушити хід схеми, текст слід розмістити на окремому аркуші і дати перехресну виноску на символ.

У схемах може використовуватися **ідентифікатор символів**. Це зв'язаний з даним символом ідентифікатор, котрий визначає символ для

використання як довідки в інших елементах документації (наприклад, у лістингу програми). Ідентифікатор символу повинен міститися ліворуч над символом.

У схемах може використовуватися **опис символів** – будь-яка інша інформація, наприклад, для відображення спеціального призначення символу з перехресною виноскою, або двома виносками для кращого розуміння функції як частини схеми. Опис символу має бути розміщений праворуч над символом.

Приклад.



У схемах роботи системи символи, що відображають носії даних, у багатьох випадках представляють **способи введення-виведення**. Для використання як виноски на документацію текст на схемі для символів, що відображають способи введення, повинен розміщуватися праворуч над символом, а текст для символів, що відображають способи виведення – праворуч під символом.

Приклад.

3.5. Правила виконання з’єднань

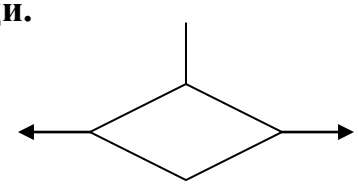
Потоки даних або потоки керування в схемах показуються лініями. Напрям потоку зліва направо і згори донизу вважаються стандартними.

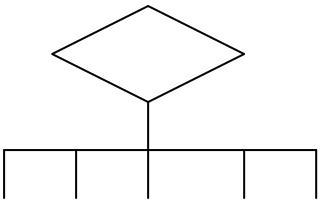
Кілька виходів із символу слід показувати:

1) кількома лініями від даного символу до інших символів;

2) однією лінією від даного символу, яка потім розгалужується на відповідну кількість ліній.

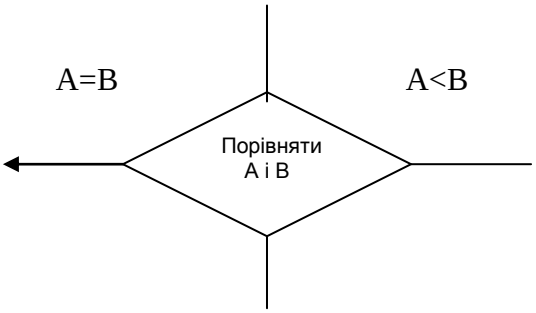
Приклади.

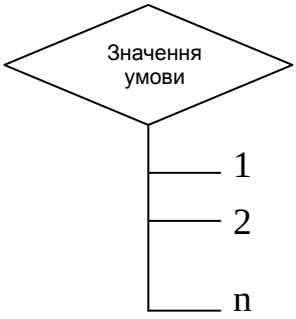




Кожен вихід із символу має супроводжуватися відповідними значеннями умов, щоб показати логічний шлях, який він представляє, з тим, щоб ці умови і відповідні виноски були ідентифіковані.

Приклади.





4.ТИПИ АЛГОРИТМІЧНИХ ПРОЦЕСІВ І ПРИНЦИПИ ЇХ ПОБУДОВИ

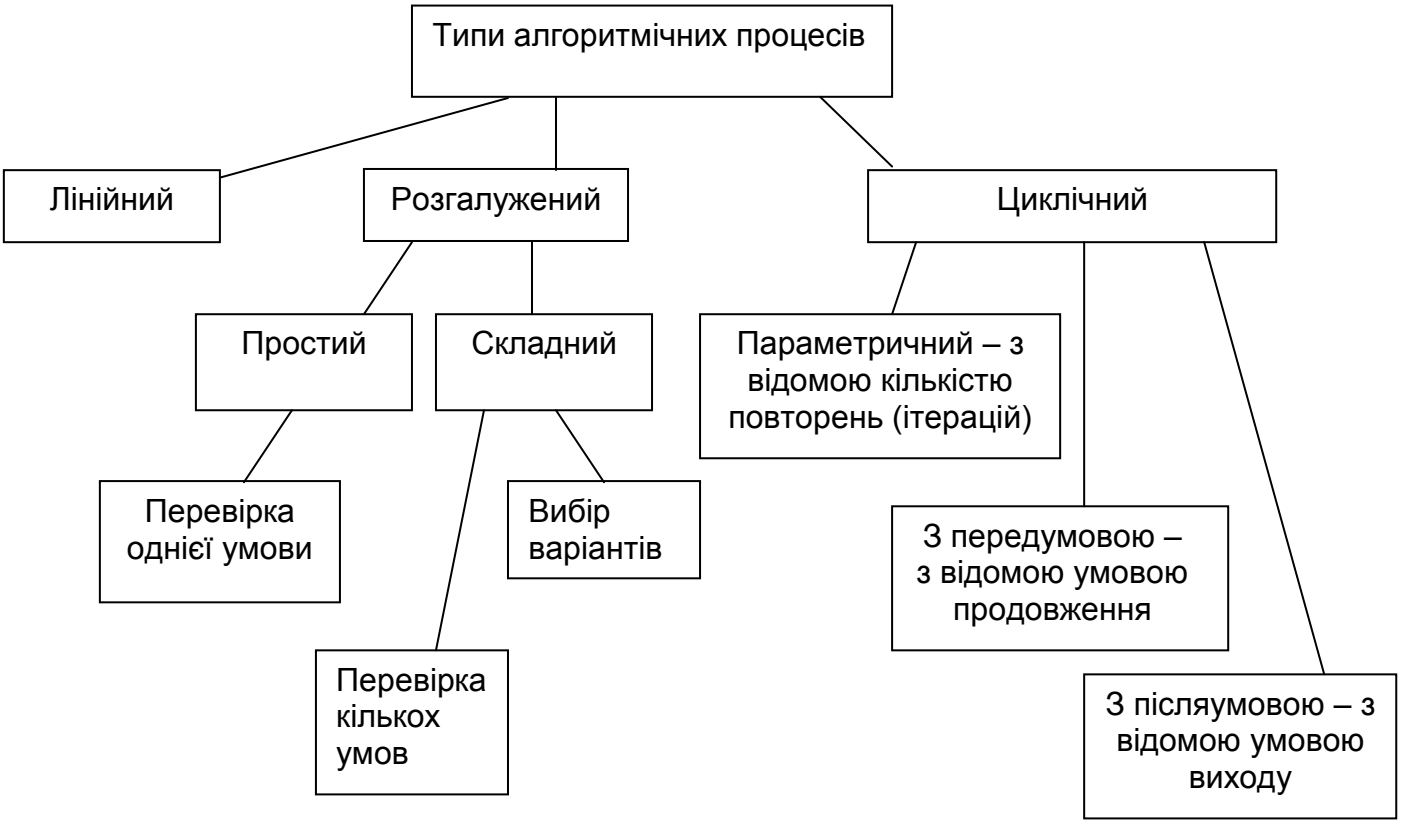


Рис.3. Типи алгоритмічних процесів

4.1.Лінійний алгоритм

Алгоритм, у якому перетворення інформації відбувається послідовно за певними формулами, які розкладаються на елементарні операції, потрібно лише визначити раціональну послідовність цих операцій (Рис.4).

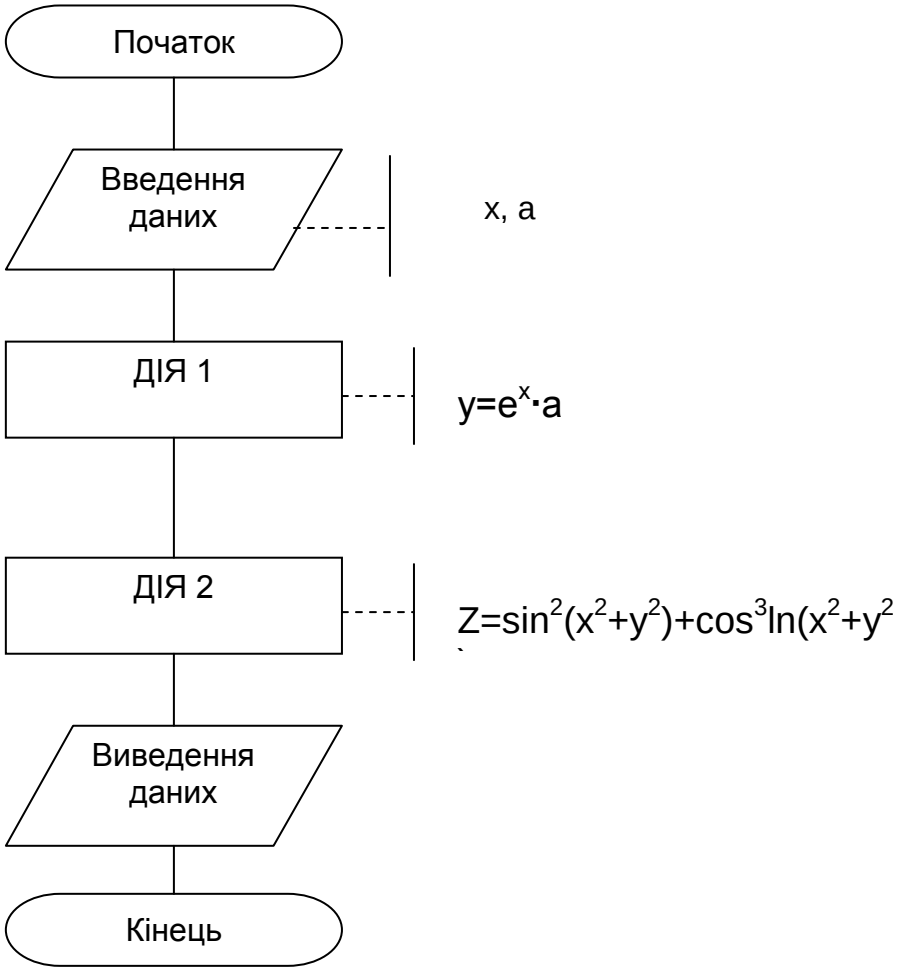


Рис.4. Приклад обчислення $Z=\sin^2(x^2+y^2)+\cos^3\ln(x^2+y^2)$, де $y=e^x \cdot a$

4.2. Розгалужений алгоритм

Алгоритм розгалуженої структури – алгоритм, у якому обчислювальний процес проходить по одній з можливих гілок, залежно від виконання деякої логічної умови (рис.5).

Розгалужені алгоритми використовуються, коли перетворення інформації може здійснюватись за різними схемами, залежно від властивостей вхідних даних або проміжних результатів.

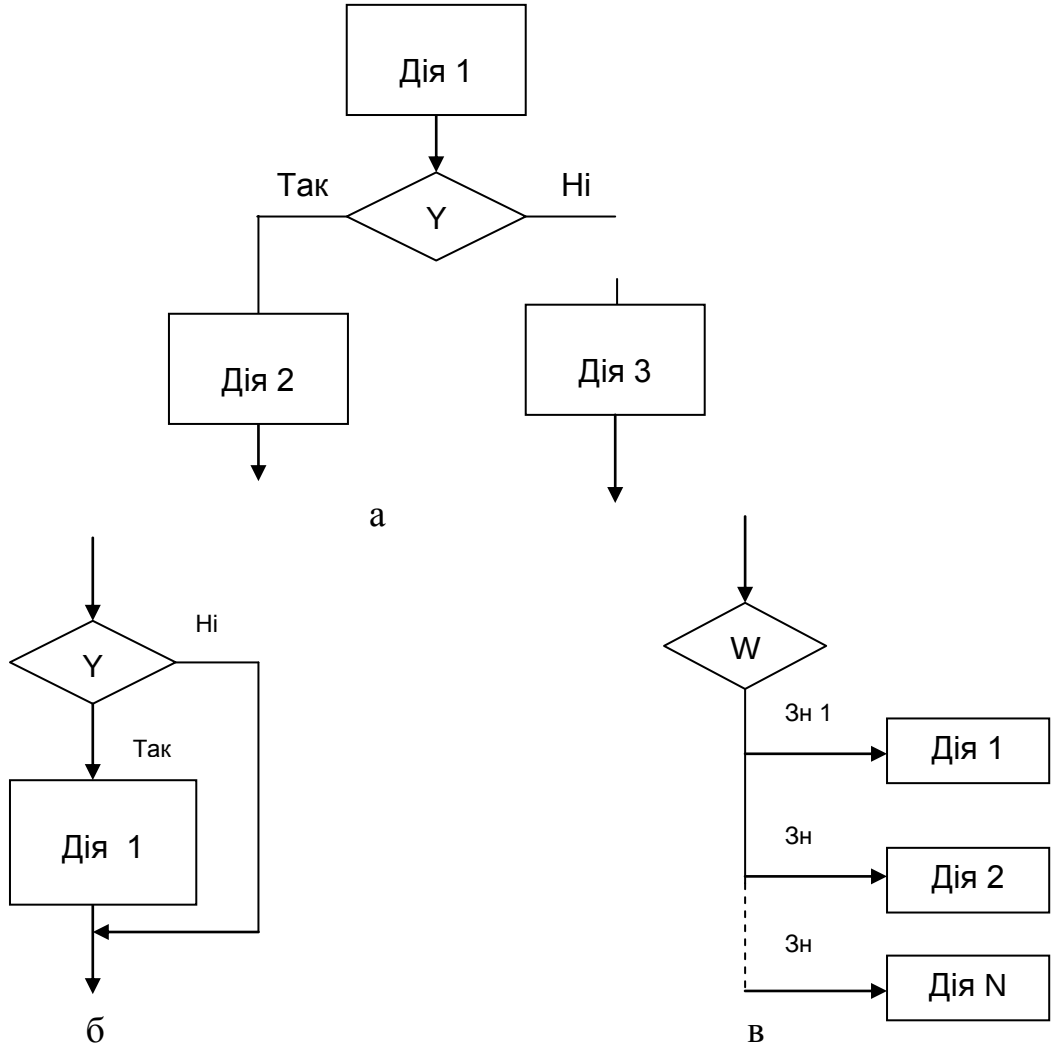


Рис.5. Схеми розгалуженого алгоритму

Тобто в алгоритмі передбачаються всі можливі варіанти обробки інформації, на кожен з яких розробляється окрема гілка алгоритму. А вибір однієї з них для виконання здійснюється за допомогою перевірки деякої умови, що відображає властивості інформації, яка використовується в процесі перетворення. В алгоритмічних системах для цього є розпізнавачі.

Залежно від того, на скільки гілок розгалужується алгоритм, він може бути простий або складний. Для простого розгалуженого процесу перевіряється одна умова (один розпізнавач), для складного – дві чи більше умов, кожна з яких відокремлює одну гілку.

Прості розгалуження показано на рис.5 а) та б). Умова формулюється так, щоб відповідь перевірки була “Так” чи “Ні” (рис.6.).

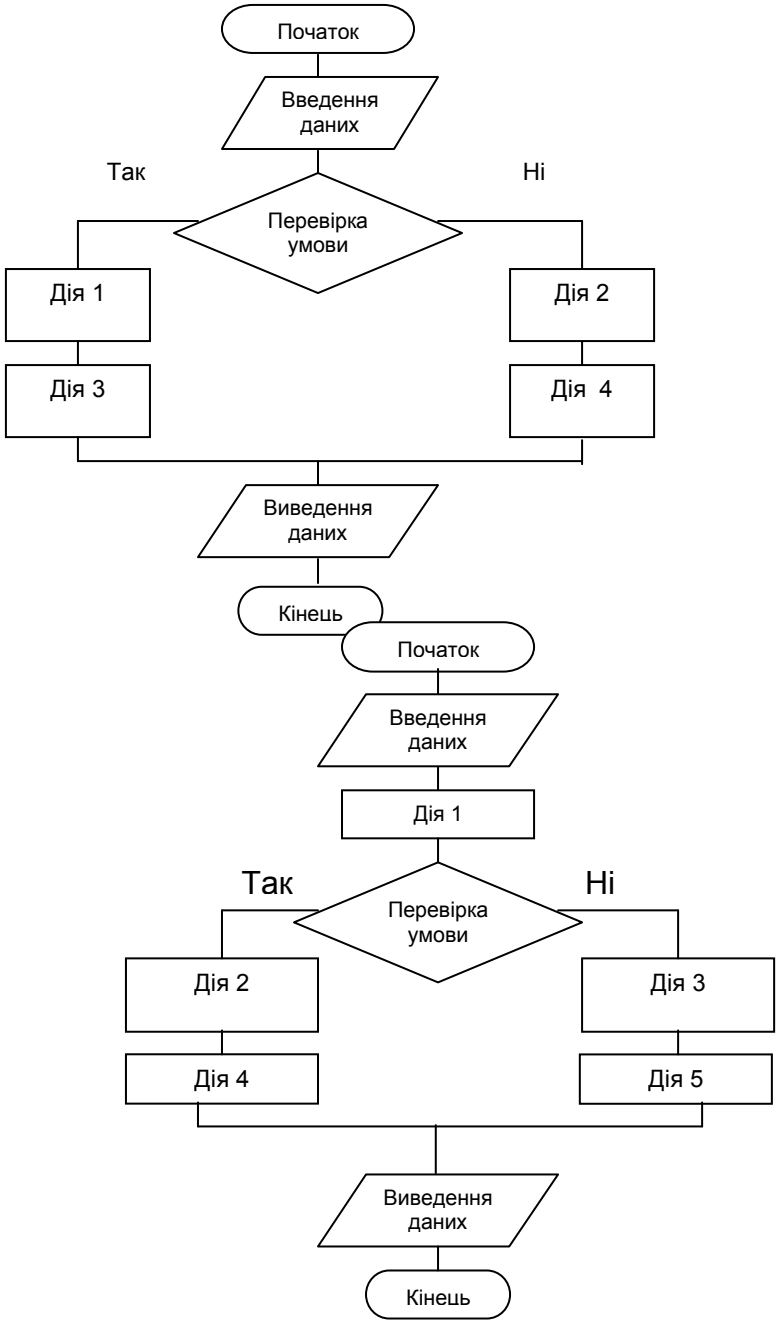


Рис. 6. Загальні схеми алгоритму розгалуженого процесу.

Наведемо приклади простих розгалужених алгоритмів.

Приклад 1. Дано числа a та b. Обчислити max(a,b). Схему алгоритму розв’язання показано на рис. 7:

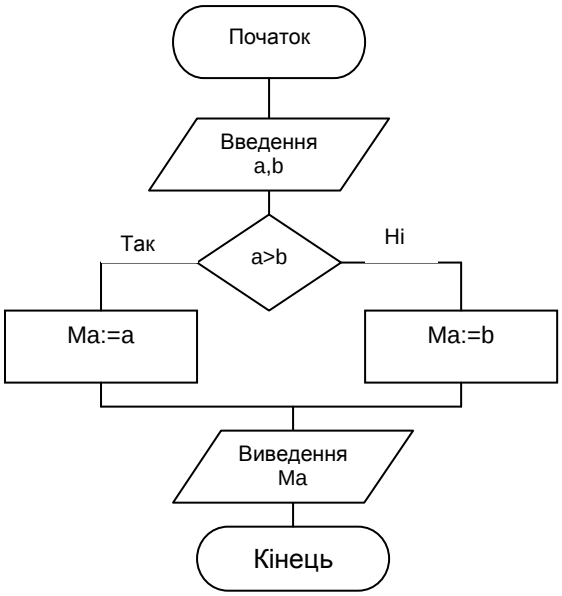


Рис. 7. Схема алгоритму обчислення максимуму з двох значень.

Приклад 2. Дано дійсні числа a,b,c. Обчислити min(a+b+c,abc) (рис. 8).

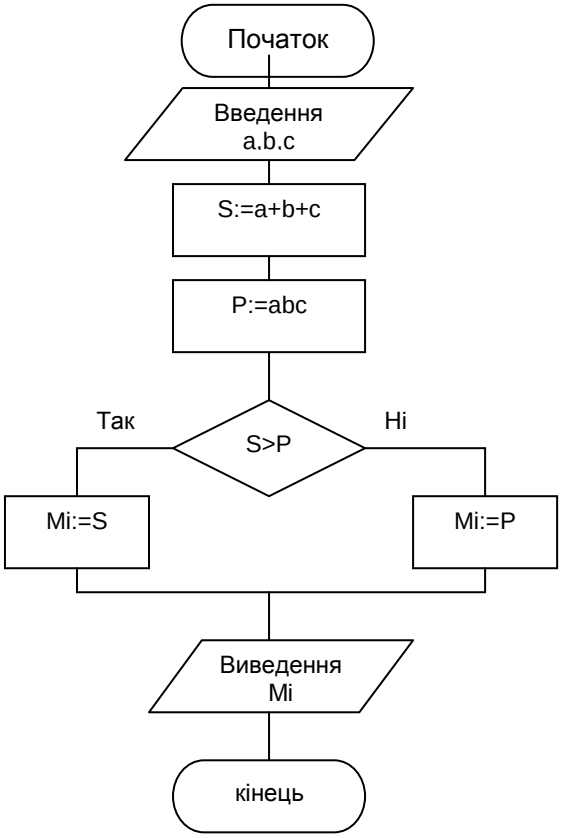


Рис. 8.Схема алгоритму розв’язання

Складне розгалуження відбувається послідовно з відокремленням гілок за діями (рис. 9).

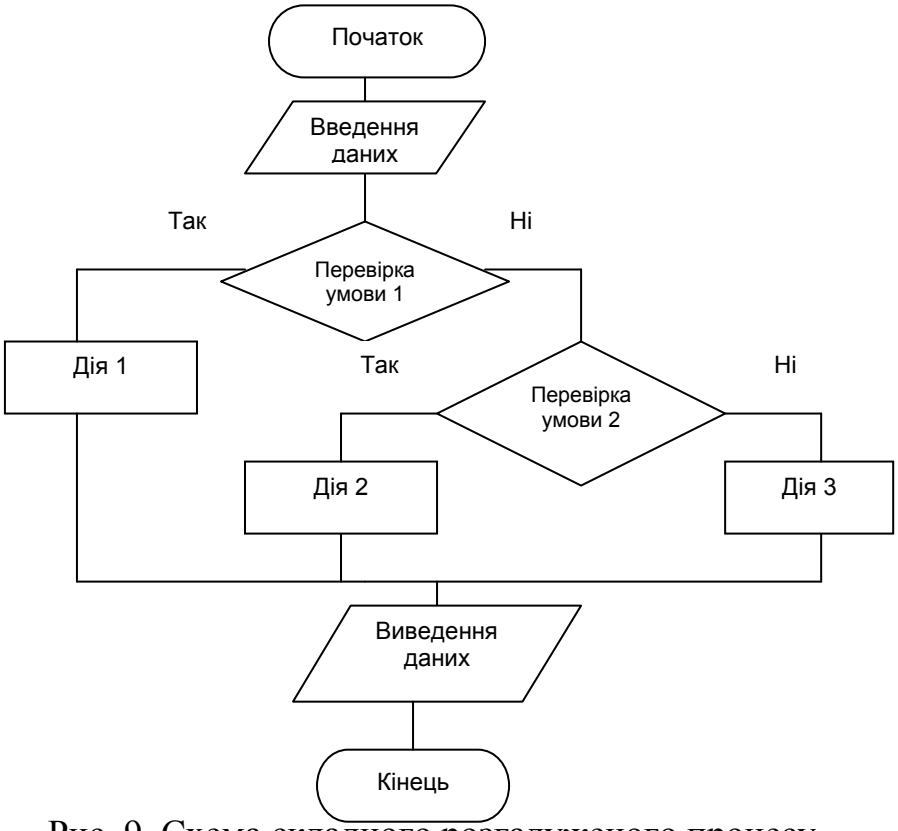


Рис. 9. Схема складного розгалуженого процесу.

4.3. Циклічні алгоритми

Циклом називається певна послідовність неодноразово повторюваних дій.

Характерною особливістю алгоритмів циклічних структур є наявність **замкнутого контуру**. Послідовність дій у контурі створюють **тіло циклу.** Одне виконання тіла циклу називається *ітерацією.*

Для побудови циклічного алгоритму необхідно:

визначити всі дії, які потрібно виконати до входу в цикл, тобто провести підготовку циклу;

визначити всі операції, які складатимуть тіло циклу;

вибрати найбільш відповідний вид циклу.

Існують такі структури циклу: з параметром; з передумовою; з післяумовою; вкладений цикл.

4.3.1. Цикл з параметром

Такий цикл використовують, коли наперед відома кількість ітерацій циклу. Алгоритм побудови такого циклу подають у вигляді схеми з використанням блоку модифікації параметра циклу. Параметр циклу – це величина, яка змінює своє значення від початкового до кінцевого на один порядок у бік збільшення або зменшення.

Параметричний цикл показано на рис.10.

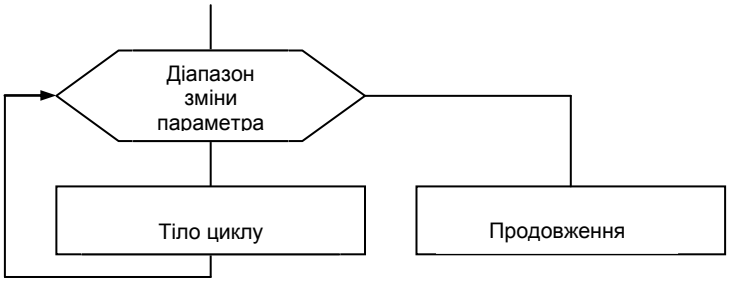


Рис.10. Загальна схема параметричного циклу

Алгоритм роботи циклу з параметром:

1. Параметру циклу присвоюється початкове значення.
2. Перевіряється, чи воно не перевищує кінцевого значення.
3. Якщо не перевищує, то виконується тіло циклу, якщо перевищує, то здійснюється перехід до п.6.
4. До параметра циклу додається крок (величина певного порядку).
5. Перехід до п. 2.
6. Вихід з циклу.

Приклад 3. Скласти схему алгоритму обчислення суми послідовності чисел ряду від 1 до n: S=1+2+..+n.

Розв'язання. Для накопичення суми використовуємо простий циклічний алгоритмічний процес, в якому параметр циклу „i" буде змінювати значення від 1 до n з кроком 1 (рис.11). Перед початком циклу задамо значення суми S=0. У тілі циклу слід передбачити накопичення суми: S+i. Умовою виходу з циклу є перевірка: чи i>n?

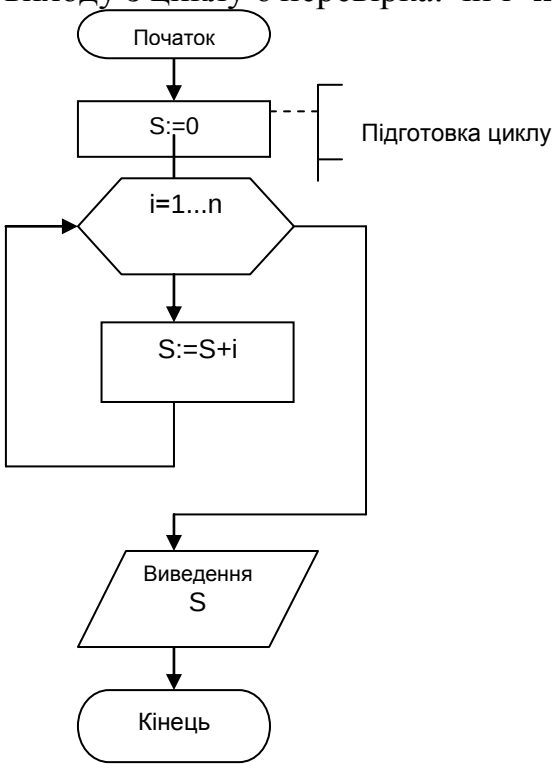


Рис. 11. Схема алгоритму розв'язання прикладу 3

Приклад 4

Обчислити середнє арифметичне значення додатніх значень функції y, якщо відомий проміжок зміни аргументу x та крок, з яким змінюється x.

$y=\ln x/(\sin^2 x+1/x)$, $x \in [5;10]$; $\Delta x=0.1$;

Розв'язання. Позначимо як k – кількість додатніх значень функції y, як S – суму додатніх значень функції y, як n – кількість значень функції y на заданому проміжку, як $x_п$ – початкове значення аргумента x, x_k – кінцеве значення аргумента x, Sa – середнє арифметичне значення додатніх значень функції y. Схема алгоритму розв'язання прикладу наведена на рис. 12. Цикл організований за кількістю значень обчислюваної функції. У тілі циклу обчислюється поточне значення функції та порівнюється з нулем. За додатнім значенням функції відбувається накопичення суми та збільшення кількості додатніх значень.

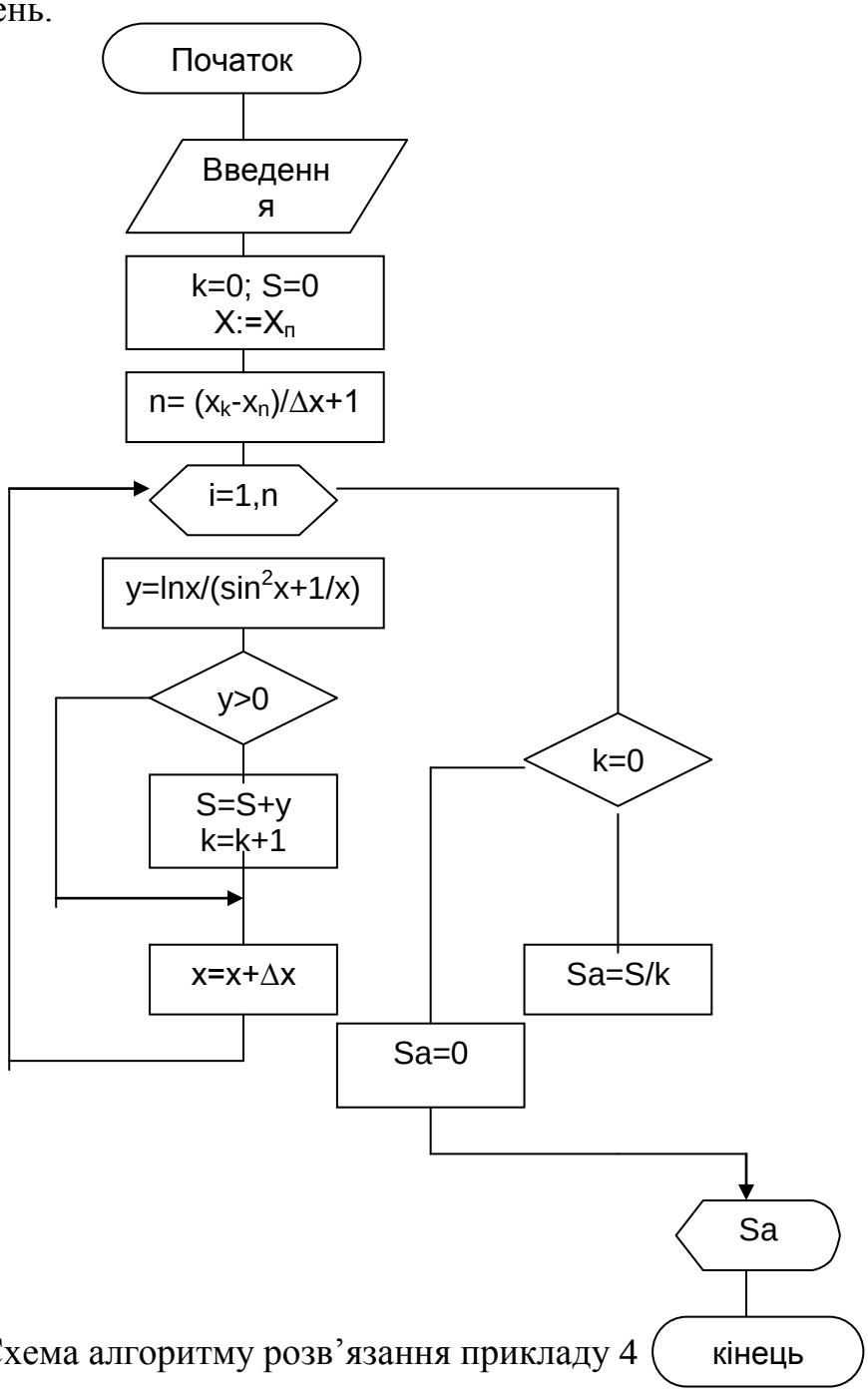


Рис. 12. Схема алгоритму розв'язання прикладу 4

3.3.2. Цикл з передумовою

Це такий цикл, у якому умова передує тілу циклу і називається умовою продовження циклу. Його слід використовувати у разі, якщо наперед невідома кількість ітерацій, але відома умова продовження циклу. Цикл з передумовою показано на рис.13.

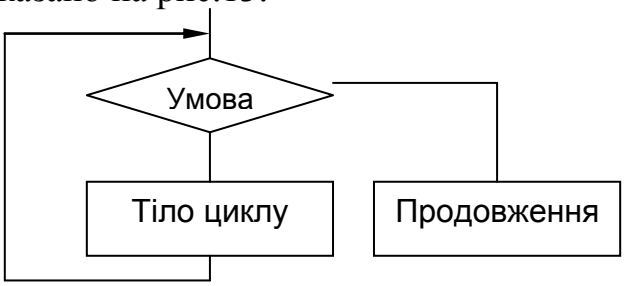


Рис.13. Схема циклу з передумовою

Алгоритм роботи циклу з передумовою:

1. Обчислюється блок умови.
2. Якщо значення “Так”, то виконується тіло циклу, якщо “Ні”, то здійснюється перехід на продовження програми.

Приклад 5. Скласти схему алгоритму обчислення суми послідовності чисел ряду від 1 до n: S=1+2+..+n з використанням алгоритму циклу з передумовою.

Розв’язання. Схему алгоритму розв’язання задачі показано на рис. 14.

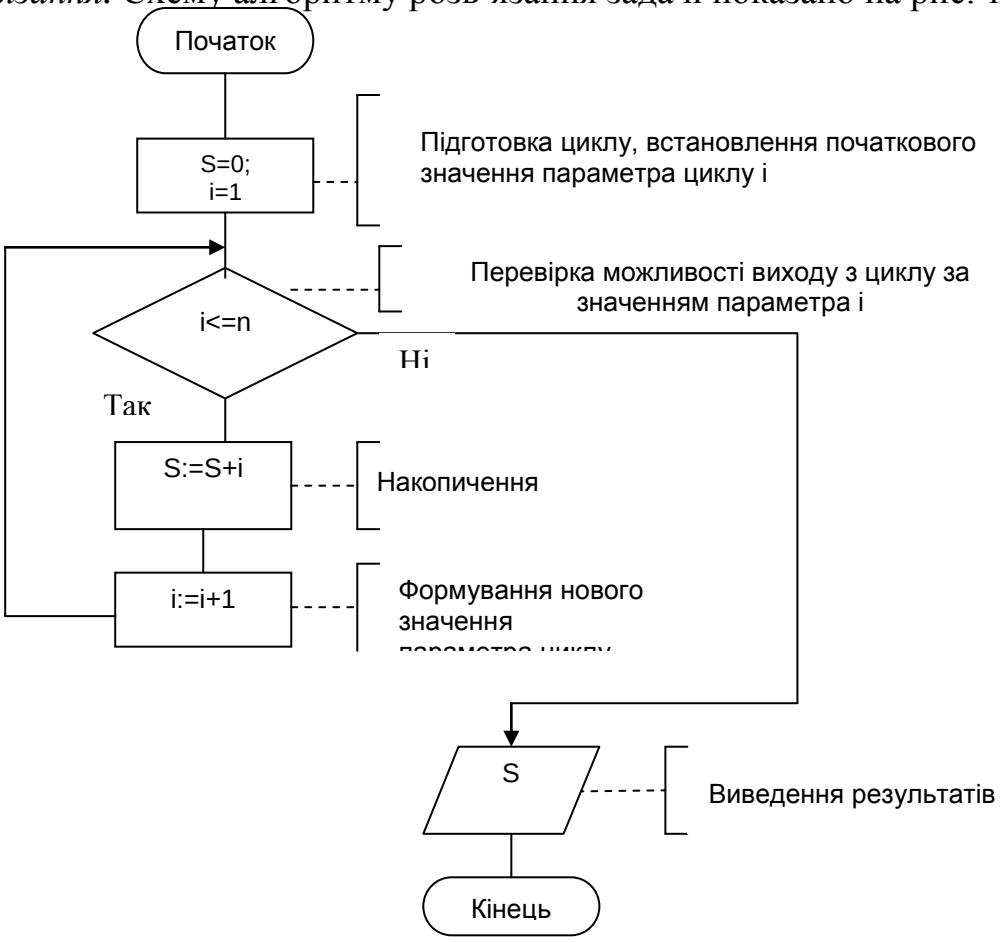


Рис. 14. Схема алгоритму розв’язання прикладу 5

4.3.3. Цикл з післяумовою

Це такий цикл, у якому умова вказується після тіла циклу і є умовою виходу з циклу. Його слід використовувати у разі, якщо наперед невідома кількість ітерацій, але відома умова виходу з циклу. Цикл з післяумовою показано на рис.15.

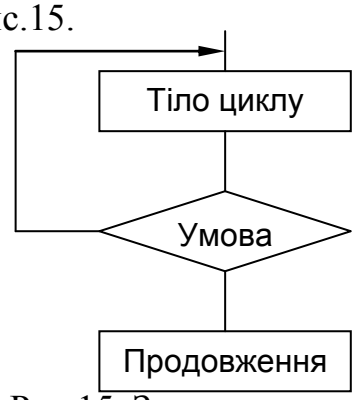


Рис.15. Загальна схема циклу з післяумовою

Алгоритм роботи циклу з післяумовою:

1. Виконується тіло циклу.
2. Обчислюється блок умови.
3. Якщо значення “Так”, то здійснюється вихід з циклу, тобто виконується перехід на продовження програми, якщо “Ні”, то виконується наступна ітерація тіла циклу.

Особливістю даного циклу, на відміну від інших, є те, що тіло циклу виконується завжди принаймні 1 раз.

Приклад 6. Скласти схему алгоритму обчислення суми послідовності чисел ряду від 1 до n: S=1+2+..+n з використанням алгоритму циклу з післяумовою.

Схему алгоритму розв’язання задачі показано на рис. 16.

Алгоритми циклів з передумовою та післяумовою використовуються, коли кількість повторень у циклі невідома.

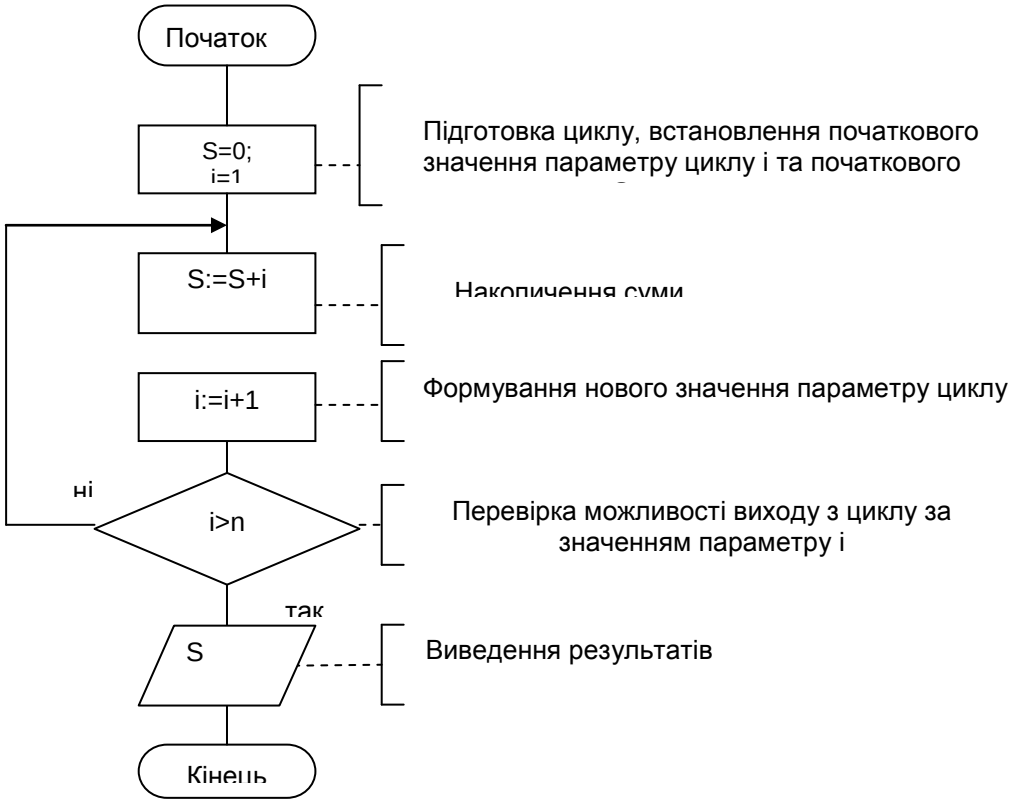


Рис. 16 Схеми алгоритмів розв'язання прикладу 6

Приклад 7. Скласти схему алгоритму обчислення суми послідовності чисел ряду $S = \sum_{i=1}^{\infty} \frac{1}{i^2}$.

Обчислення припинити, як тільки $\frac{1}{i^2} \le \varepsilon$, де ε=0.0001.

Розв'язання. Умова виходу з циклу може перевірятися до початку циклу (рис.17 а), або після виконання тілу циклу (рис.17 б).

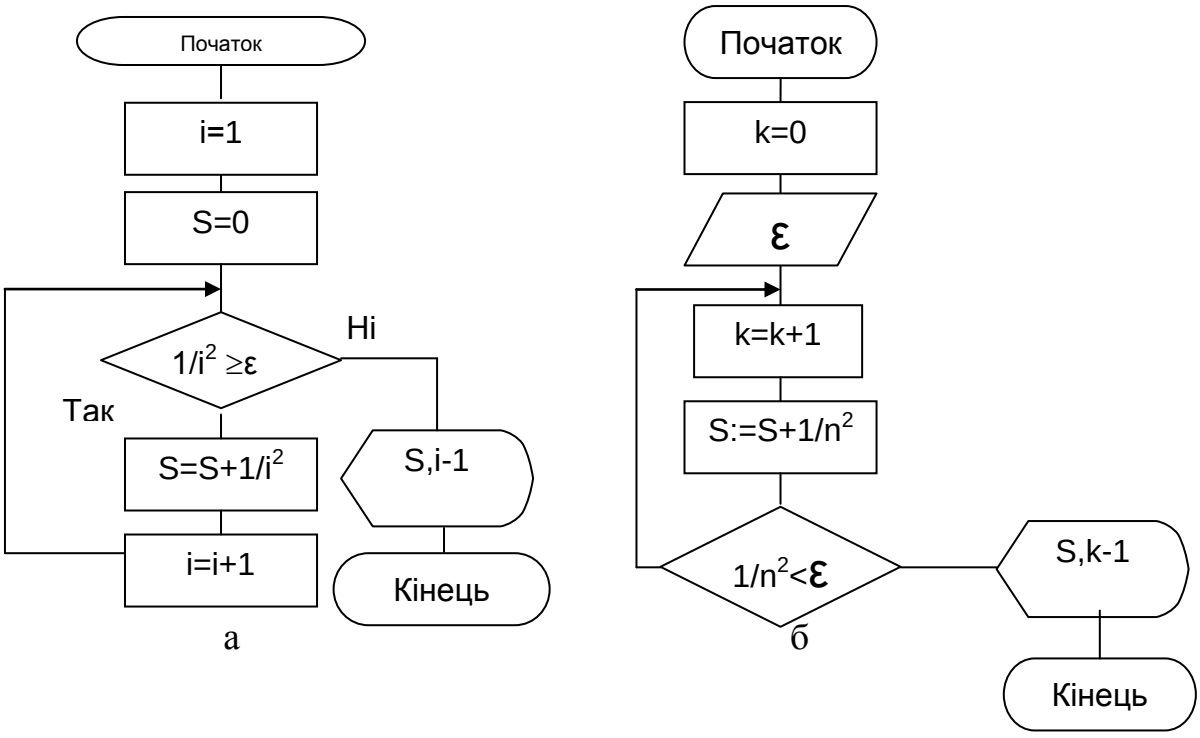


Рис.17. Схеми алгоритмів розв'язання прикладу 7

3.3.4. Вкладені цикли

Якщо один цикл повністю міститься в тілі іншого циклу, то таку конструкцію називають вкладеними циклами. На глибину вкладень обмежень не накладають. Вкладені цикли зазвичай використовуються для обробки багатовимірних масивів або ж інших алгоритмів зі складною структурою.

Приклад 8. Обчислити значення Z:

$$Z = \sum_{i=1}^{15} \frac{i}{(i+2)!}$$

Розв'язання. Організуємо зовнішній цикл для обчислення суми, а внутрішній цикл для обчислення факторіалу. Алгоритм розв'язання задачі наведений на рис. 18.

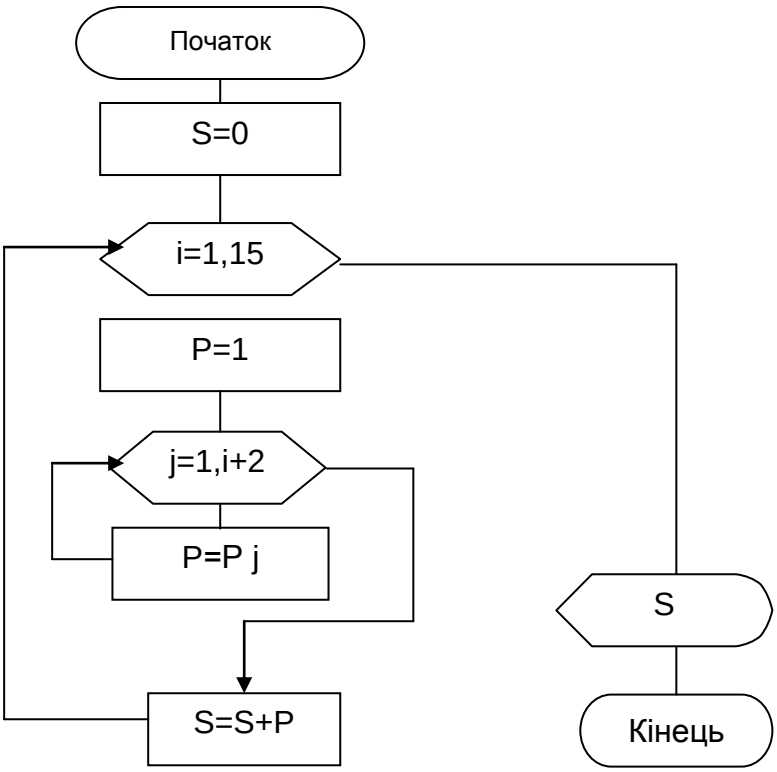


Рис.18. Схема алгоритму розв’язання прикладу 8

ЛІТЕРАТУРА

Основна

1. *Глинський Я.М*. ІНФОРМАТИКА: 8–11 класи: Навч. посібник для загальноосвіт. навч. закладів: [У 2-х кн.] – Кн.1. Алгоритмізація і програмування. Мова Паскаль. – 2–е вид. – Л.: Деол, 2002. – 2000с.
2. *Информатика*. Базовый курс / С.В. Симонович и др.– СПб.: Питер, 1999. –640 с.
3. *Марченко А.И., Марченко Л.А.*. Программирование в среде Turbo Pascal 7.0: Учеб. пособие.–М.: Бином универсал; К.: Юниор, 1997. –495 с.
4. *Фаронов В.В*. Turbo Pascal 7.0. Начальний курс: Учеб. пособие. – К.: Нолидж, 2000. – 610 с.

Додаткова

5. *Методичні* вказівки до виконання лабораторних робіт в інтегрованому середовищі Turbo Pascal. – К.: УДУХТ, 1998. – 10 с. № 5292
6. *Сердюченко В.Я.* Розробка алгоритмів та програмування мовою Turbo Pascal: Навч. посібник для техн. вузів. – Х.: ВКП “Парітет” ЛТД, 1995. – 352 с.