

Міністерство освіти і науки України
Державний вищий навчальний заклад
“Ужгородський національний університет”
Математичний факультет
Кафедра системного аналізу і теорії оптимізації

ОСНОВИ АЛГОРИТМІЗАЦІЇ

Методичні вказівки до лабораторних робіт для студентів I-го курсу математичного факультету спеціальності "прикладна математика"

Ужгород – 2015

Семйон І.В., Чупов С.В., Брила А.Ю., Антосяк П.П., Дудла М.В. Основи алгоритмізації. Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності "прикладна математика". – Ужгород, 2015. – 63с.

Розглядаються основні поняття алгоритмізації: означення, способи запису, типи алгоритмів, структурний підхід до розробки алгоритмів та допоміжні алгоритми.

Для кожної із тем наведено необхідні теоретичні відомості та тестові завдання для оцінювання знань.

Методичні вказівки можуть бути використані студентами різних напрямків підготовки, а також для профільного вивчення інформатики у школі.

Рекомендовано до друку Вченою радою математичного факультету

ДВНЗ “Ужгородський національний університет”

від 21 травня 2015 року, протокол №10.

1. ТЕОРЕТИЧНІ ОСНОВИ

1.1. *Поняття алгоритму*

У своїй повсякденній діяльності нам постійно приходиться стикатися з різного роду правилами, що вказують на послідовність дій, виконання яких приведе до очікуваного результату. Таких правил дуже багато. Наприклад, ми маємо діяти за цілком визначеною системою правил, щоб подзвонити по телефону чи пройти в метро, обчислити добуток багатозначних чисел чи знайти корені квадратного рівняння. Необхідно виконати певну послідовність дій, щоб зварити кашу або піднятися в ліфті на потрібний поверх або за допомогою циркуля і лінійки поділити відрізок навпіл. Прикладів такого роду послідовностей дій можна навести чимало. Їх часто називають інструкцією для виконання певної роботи. Такі описи послідовності дій зараз часто також називають алгоритмами. Однак, не всякий такий опис є алгоритмом.

Опис стає алгоритмом тільки тоді, якщо його вказівки мають певні властивості.

Потреба у більш чіткому формулюванні поняття алгоритму виникає при розгляді правил виконання арифметичних дій та геометричних побудов. Не буде грубою помилкою сказати, що елементарна (і не тільки елементарна) математика в значній мірі зводиться до відшукування алгоритмів для знаходження розв'язків задач.

Назва “*алгоритм*” пов'язана з іменем видатного математика давнини *Мухаммеда бен-муса аль-Хорезмі* (IX ст. до н. е.). Він виклав загальні правила виконання арифметичних дій над числами, представленими в десятковій системі числення. Цими правилами ми користуємося й досі.

До появи цих правил існувало чимало способів виконання арифметичних операцій, де враховувались ті чи інші особливості конкретних

чисел. Наприклад, якщо один із співмножників (нехай перший) є число, що складається тільки з дев'яток тоді, до другого співмножника треба дописати число нулів, рівне числу дев'яток в першому, а потім від отриманого відняти другий співмножник. Якщо число 999 треба помножити на 5, ми повинні (за вказаним правилом) до числа 5 дописати три нулі (оскільки в першому числі є три дев'ятки), одержимо – 5000 і далі від нього відняти другий множник, тобто число 5. Результатом буде число 4995, яке і є добутком чисел 999 та 5.

Багатьма прийомами усного рахунку ми користуємося й зараз. Наприклад, щоб помножити будь-яке число на 5, необхідно, дописати до нього нуль і поділити отримане число на 2. Або, щоб помножити на 25 необхідно до даного числа дописати два нулі і отримане поділити на 4. Та подібними прийомами можна користуватися лише у випадку, коли один із співмножників має спеціальний вигляд (у першому прикладі це має бути число виду $99\dots9$, у інших – 5 і 25).

Аль-Хорезмі запропонував правила придатні для всіх випадків і однакові для будь-якої пари чисел. Прихильників аль-Хорезмі почали називати “алгоритміками”, а під словом “алгоритм” стали розуміти систему правил, що має певні властивості.

Поняття алгоритму тісно пов'язане з поняттям виконавця алгоритму. Виконавцем алгоритму може бути людина, яка може виконати всі вказівки алгоритму. Для людини вказівки алгоритму повинні бути такими, щоб людина їх могла виконати. Якщо алгоритм має виконувати ЕОМ, то вказівки мають бути такими, щоб їх міг виконати процесор. Тобто, кожний виконавець характеризується деякою сукупністю дій (команд, інструкцій), які може виконати цей виконавець.

Зауважимо, що дії тої чи іншої інструкції передбачають наявність об'єктів над якими ця дія виконується. Кожний алгоритм передбачає наявність деяких вхідних (початкових) даних, які ще називають *аргументами* та отримання певного результату. Так, для виконання арифметичної операції

над двома числами вхідними даними будуть пари чисел, результатом же операції – одне число.

При розробці алгоритмів окрім аргументів та результатів можна виділити і так звані *проміжкові величини*, що використовують для збереження проміжкових даних. Так, наприклад, для обчислення площі трикутника за довжинами його сторін, площа є результатом, довжини сторін – аргументами, а півпериметр, який використовується у формулі Герона, є проміжковою величиною.

Алгоритм – це опис послідовності інструкцій (команд, вказівок) для певного виконавця, виконання яких за скінчену кількість кроків приводить до отримання результату для довільного допустимого набору вхідних даних.

Оскільки алгоритм розрахований на певного виконавця, то кожна вказівка алгоритму має бути елементарною, тобто такою, яка може бути виконана цим виконавцем. Це одна з найважливіших властивостей вказівок алгоритму. Другою важливою властивістю вказівок алгоритму є їх визначеність (детермінованість), яка полягає в тому, що застосування цієї вказівки для однакових вхідних даних має давати один і той же результат. Алгоритм повинен мати також вказівку початку та вказівку кінця.

Інші властивості алгоритмів:

Дискретність. Властивість дискретності означає покроковий характер процесу його виконання де кожний крок має бути відокремлений. Тобто ми чітко можемо вказати де закінчується один крок і починається наступний.

Масовість. Зміст масовості алгоритму полягає в тому, що алгоритм придатний для розв’язання цілого класу задач, і для кожної окремої задачі з цього класу знаходить її розв’язок.

У свій час, противники методів аль-Хорезмі зневажливо ставились до “алгоритміків” саме через те, що вони зробили доступним і зрозумілим вміння виконувати арифметичні дії, розвінчавши загадковість “мистецтва обчислень”. Адже виконання таких дій дозволило виконувати арифметичні дії всім хто міг виконувати вказівки алгоритму.

Однозначність. Алгоритм є *однозначним*, якщо при застосуванні до одних й тих самих даних він дає один й той самий результат. Так, застосовуючи алгоритм для множення однакових пар чисел, завжди отримаємо той же самий результат. І якщо при цьому порівнюватимемо результати, отримані після кожного відповідного кроку алгоритмічного процесу, то виявиться, що при однакових вхідних даних проміжкові результати також будуть однаковими.

Визначеність. Кожний крок алгоритму має бути чітко і однозначно визначений, щоб не допускати довільного трактування виконавцем. Як уже згадувалось раніше, алгоритм розрахований на механічне виконання. Якщо один і той самий алгоритм доручити для виконання різним виконавцям, то вони повинні отримати один і той самий результат.

Зрозумілість. Наступна важлива властивість, якою має володіти алгоритм – це *зрозумілість* для виконавця, тобто виконавець алгоритму знає як його виконувати. Формулювання дій алгоритму повинно бути орієнтоване на конкретного виконавця, а його опис має бути настільки точним і однозначним, щоб повністю визначати усі дії виконавця.

Скінченність. Виконання алгоритму припиняється після завершення скінченної кількості кроків.

Результативність – за скінченну кількість кроків алгоритм має приводити до розв'язання задачі, або зупинятися через неможливість її виконання.

Була б хибною думка про те, що для певної задачі існує лише один алгоритм її розв'язку. Як правило таких алгоритмів є багато і називаються вони *еквівалентними*. В цьому випадку, зрозуміло, постає питання визначення ефективності цих алгоритмів. Так виник новий розділ теорії алгоритмів – *теорія складності алгоритмів*. *Складність алгоритму* – це кількісна характеристика, яка визначається часом, за який виконується алгоритм (часова складність), та об'ємом пам'яті комп'ютера, необхідним для реалізації цього алгоритму (ємкісна складність). Очевидно, що серед

множини еквівалентних алгоритмів намагаються вибирати ті, які є найбільш ефективними для виконавця.

Отже, для деяких задач існує декілька алгоритмів їх розв'язання, а для інших задач не існує жодного і, на кінець, є задачі, для яких ми не знаємо, існує алгоритм їх розв'язання чи ні.

1.2. Способи запису алгоритмів

Існують різні способи запису алгоритмів, вибір яких залежить від того, хто його записує і на якого виконавця він орієнтований: словесний, графічний (блок-схеми), за допомогою псевдокодів, формальний (мовою програмування).

Словесний спосіб запису алгоритмів

Це найбільш проста і доступна форма представлення алгоритму. Словесна форма, зазвичай, використовується для алгоритмів, орієнтованих на виконавця – людину.

Приклад. Алгоритм Евкліда обчислення найбільшого спільного дільника (НСД) двох цілих додатних чисел викладений в «Елементах».

Крок 1. Дано два цілих додатних числа. Якщо вони рівні, то перше з них і є найбільшим спільним дільником, якщо ж ні, то перейдемо до кроку 2.

Крок 2. Порівняємо два числа і виберемо більше.


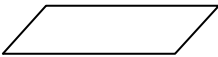


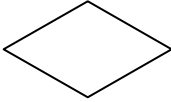
Крок 3. Більше з двох чисел замінимо різницею більшого і меншого.

Крок 4. Перейдемо до Кроку 1.

Запис алгоритмів за допомогою блок-схем

Блок-схеми дозволяють зобразити алгоритм в наочній графічній формі. Цей спосіб потребує ознайомлення з стандартами графічних зображень блоків (функціональних блоків). Деякі з цих блоків наведено в таблиці

Таблиця 2.1. Графічні зображення блоків

	<p>Початок і кінець алгоритму</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; border-radius: 15px; padding: 5px 20px; text-align: center;">Початок</div> <div style="border: 1px solid black; border-radius: 15px; padding: 5px 20px; text-align: center;">Кінець</div> </div>
	<p>Введення даних. У цьому блоці вказують величини, що є аргументами, тобто необхідні для виконання алгоритму.</p>
	<p>Виведення даних. У цьому блоці вказують величини, що є результатами.</p>
	<p>Виконання операцій, в результаті яких відбувається зміна значень величин. Зміна значень величин може бути здійснена з використанням оператора присвоєння « := »</p> <div style="border: 1px solid black; padding: 5px; margin: 10px auto; width: fit-content;"> <div style="border: 1px solid black; padding: 2px 10px; display: inline-block;">величина</div> := <div style="border: 1px solid black; padding: 2px 10px; display: inline-block;">вираз</div> </div>
	<p>Блок перевірки певної умови та вибір напрямку виконання алгоритму в залежності від виконання умови</p>
	<p>Виконання блоку послідовності операцій, що визначені в іншому місці і мають певне ім'я</p>

Під час створення блок-схеми алгоритму блоки, із записаними в них командами, з'єднуються між собою стрілками для визначення черговості виконання блоків алгоритму.

Оскільки блок-схеми алгоритмів в основному використовуються людиною, то команди всередині блоків записують за допомогою зрозумілої для людини нотації, яка може включати елементи математичної символіки.

Запишемо у вигляді блок-схеми алгоритм Евкліда

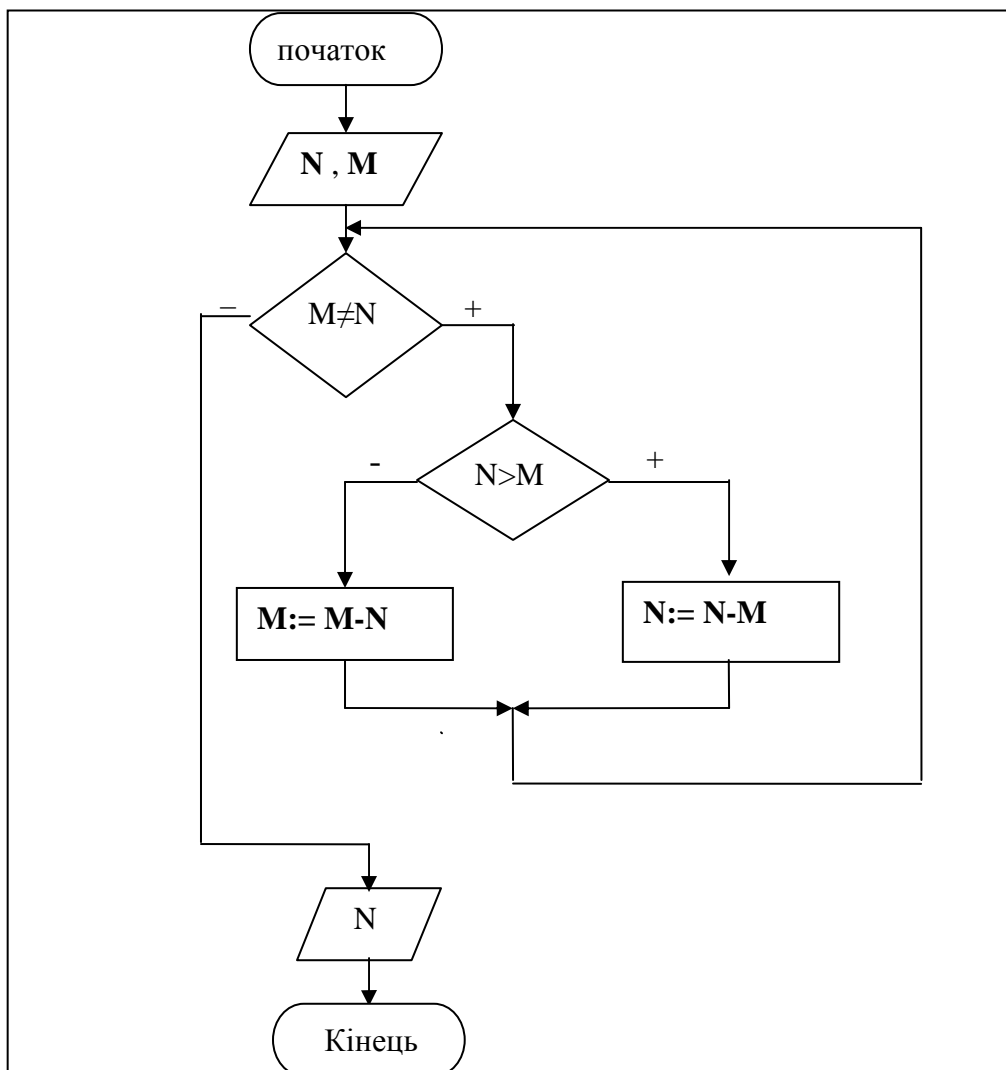


Рис. 1

Перевагою такого способу представлення алгоритму є наочність. Така форма представлення значно простіше сприймається людиною, однак ця наочність швидко втрачається, якщо кількість блоків стає достатньо великою.

Мова псевдокодів

Для запису алгоритмів за допомогою мови псевдокодів використовуються службові слова та спеціальні правила запису окремих дій. Мова псевдокодів є проміжною між природною і формальною мовами.

Розглянемо як приклад алгоритм Евкліда.

```

АЛГОРИТМ Найбільший спільний дільник
ПОЧАТОК
  ВВЕДЕННЯ «Задайте два додатних цілих числа» n,m
  ПОКИ n ≠ m
    ПОЧАТОК
      ЯКЩО n>m
        ТО n := n-m
      ІНАКШЕ m:=m-n
    ВСЕ
  КІНЕЦЬ
  ВИВЕДЕННЯ «Найбільший спільний дільник даних чисел:» n
КІНЕЦЬ

```

Перевагою такого способу опису алгоритмів є те, що службові слова є словами розмовної мови, а отже такий опис є інтуїтивно зрозумілим.

Мови програмування

Мовою програмування називатимемо фіксовану систему позначень для опису структур даних та алгоритмів, призначених для виконання обчислювальними машинами.

Приклад. Розглянемо реалізацію алгоритму пошуку найбільшого спільного дільника двох додатних цілих чисел на мові C#.

```

static void Main(string[] args)
{
    int n, m;
    Console.Write("n=");
    n = Convert.ToInt32(Console.ReadLine());
    Console.Write("m=");
    m = Convert.ToInt32(Console.ReadLine());
    while(n!=m)
    {
        if (n > m)
            n = n - m;
        else
            m = m - n;
    }
    Console.WriteLine("NSD={0}",n);
    Console.ReadLine();
}

```

1.3. Базові структури управління

Основним досягненням теорії програмування 60-х років є усвідомлення і теоретичне осмислення того факту, що існують декілька основних (базових) способів управління обчисленнями, використовуючи які можна описати будь-який алгоритм. Структура управління будь-якого алгоритму може бути реалізована у виді комбінації основних структур управління. До них відносяться: структура слідування, структури розгалуження, структури повторення.

Структура слідування

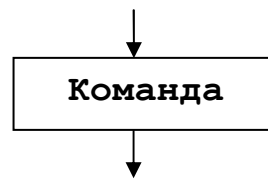


Рис. 2

За допомогою структури слідування описують процес перетворення даних, що не вимагає перевірки жодних умов чи виконання повторюваних операцій.

Структури розгалуження

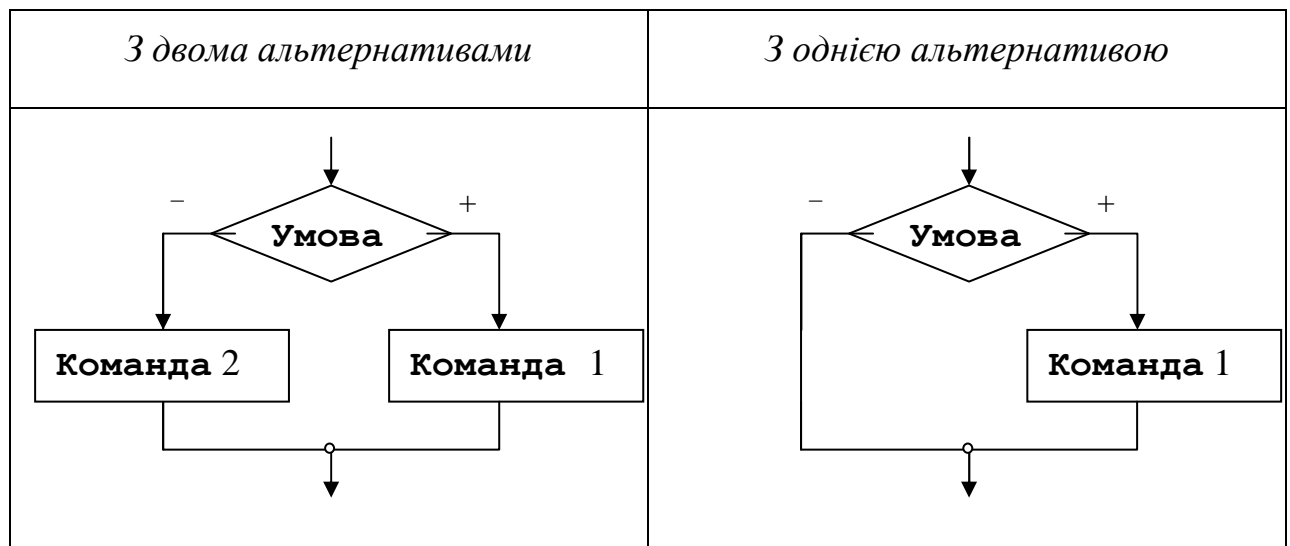


Рис. 3

Структури розгалуження дозволяють описувати випадки, коли, у залежності від виконання (+) чи невиконання (–) деякої умови, необхідно виконати одну із команд. У випадку структури розгалуження з однією

альтернативою, якщо умова виконується (+), то команда виконується, інакше – не виконується жодних дій.

Якщо умова є складною і складається з декількох, то їх можна поєднувати з використанням службових слів « і » та « або », які підкреслюють. Так наприклад, для перевірки належності x відрізьку $(a;b]$ необхідно перевірити умову

$$(a < x) \underline{і} (x \leq b).$$

Для перевірки, чи є трикутник з сторонами a, b, c рівнобедреним, необхідно перевірити умову

$$(a = b) \underline{або} (b = c) \underline{або} (a = c).$$

Для перевірки, чи є прямокутники зі сторонами a_1, b_1 і a_2, b_2 рівними, необхідно перевірити умову

$$((a_1 = a_2) \underline{і} (b_1 = b_2)) \underline{або} ((a_1 = b_2) \underline{і} (b_1 = a_2)).$$

Структури повторення

Структури повторення (циклу) використовують у випадках, коли певну послідовність команд необхідно повторювати до тих пір, поки виконується деяка умова. Виділяють структури повторення з передумовою, з післяумовою та з параметром.

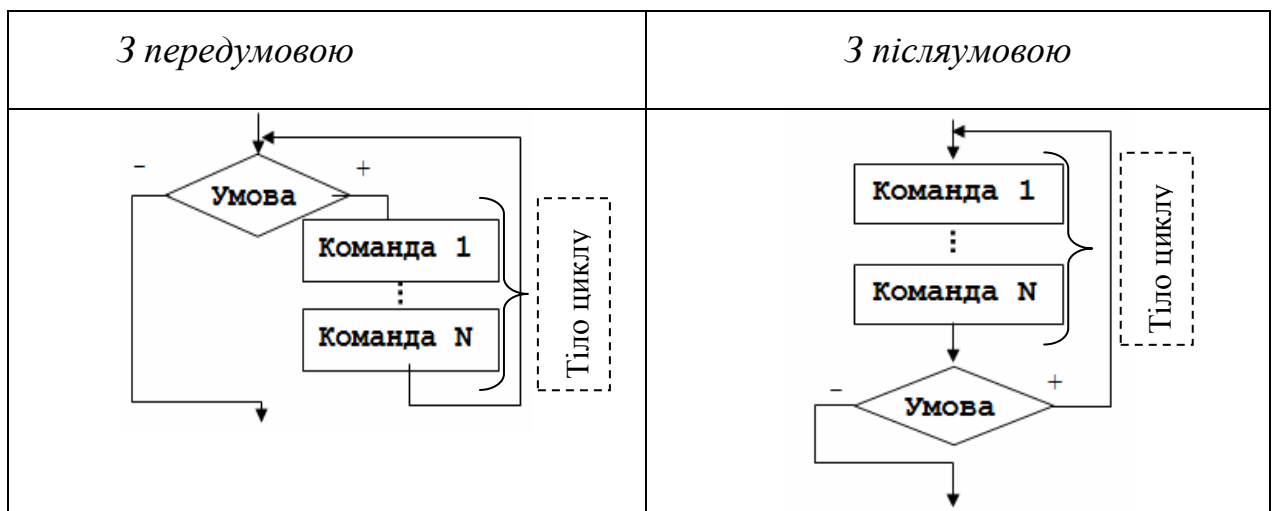


Рис.4

Команди, які необхідно повторювати у циклі, утворюють *тіло циклу*. Суттєвою відмінністю структури з післяумовою є те, що команди, які утворюють тіло циклу, обов'язково будуть виконані принаймні один раз. В той же час, при використанні структури циклу з передумовою можлива ситуація, коли команди не будуть виконані жодного разу.

Структура повторення з параметром

Структура повторення з параметром використовується у випадках, коли з кожним повтором виконання тіла циклу можна співставити деяке значення параметра, який поступово змінюється від початкового до кінцевого (включно) із заданим кроком.

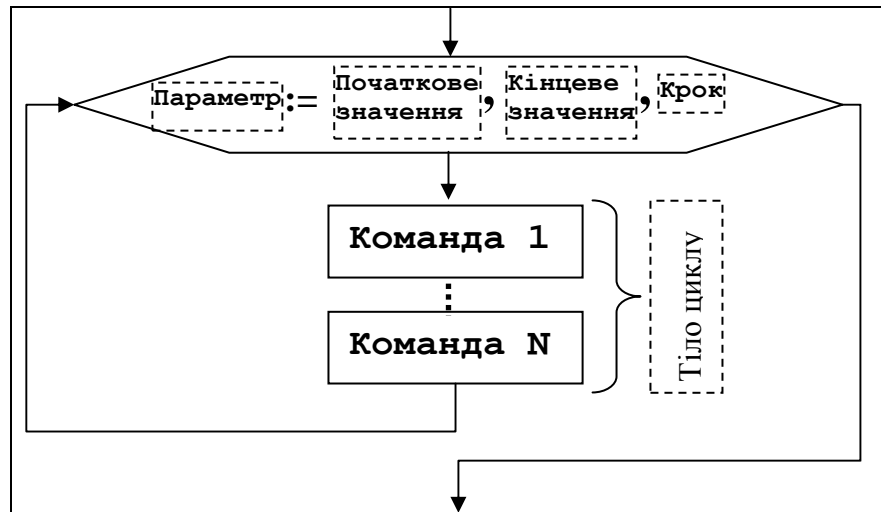


Рис.5

Цикл з параметром може бути замінений з використанням одного із циклів з передумовою, у залежності від кроку.

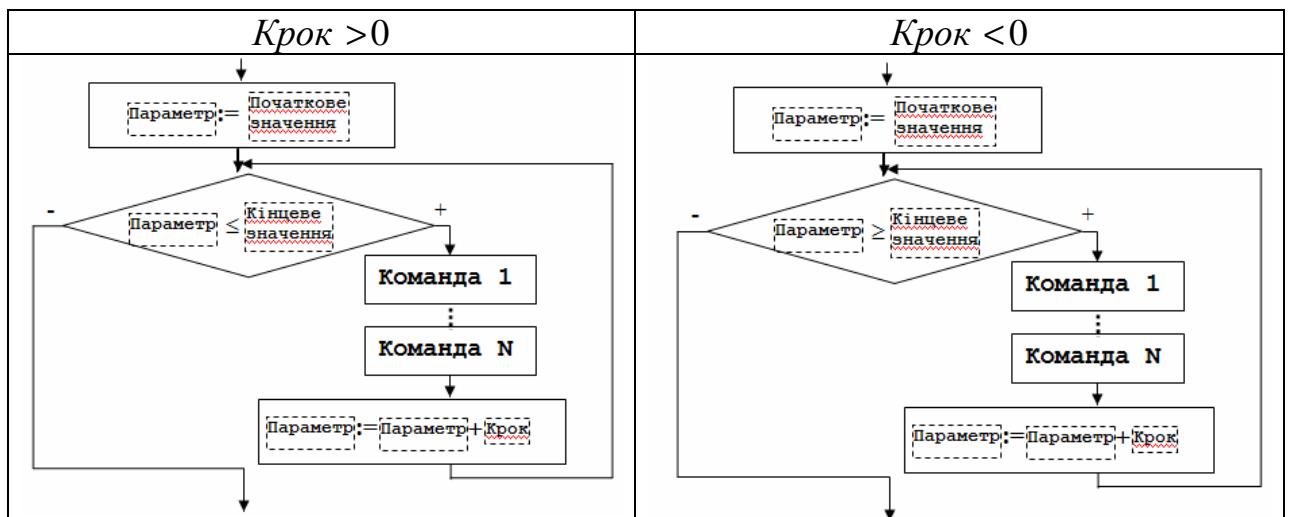


Рис.6

Зауважимо, що кожна із структур має один вхід і один вихід.

Крім описаних структур, в алгоритмах можуть використовуватися і інші структури. Хоча даних чотирьох елементарних структур цілком достатньо при розробці блок-схеми будь-якого алгоритму, але як сама схема так і її побудова не обов'язково виявляться простими, наглядними та зрозумілими. Наприклад, розглянемо випадок, коли, у залежності від значення деякої величини (селектора вибору), необхідно виконати одну із команд: якщо величина приймає значення 1 , то необхідно виконати команду 1 , якщо значення 2 – команду 2 , ..., значення m – команду m , якщо жодне із вказаних значень, то команду $m+1$. В цьому випадку доведеться використовувати певну кількість структур розгалуження вкладених одна в одну. Однак, програма з вкладеними структурами розгалуження стає менш наочною і зрозумілою. Тому часто при розробці алгоритмів використовують структуру вибору.

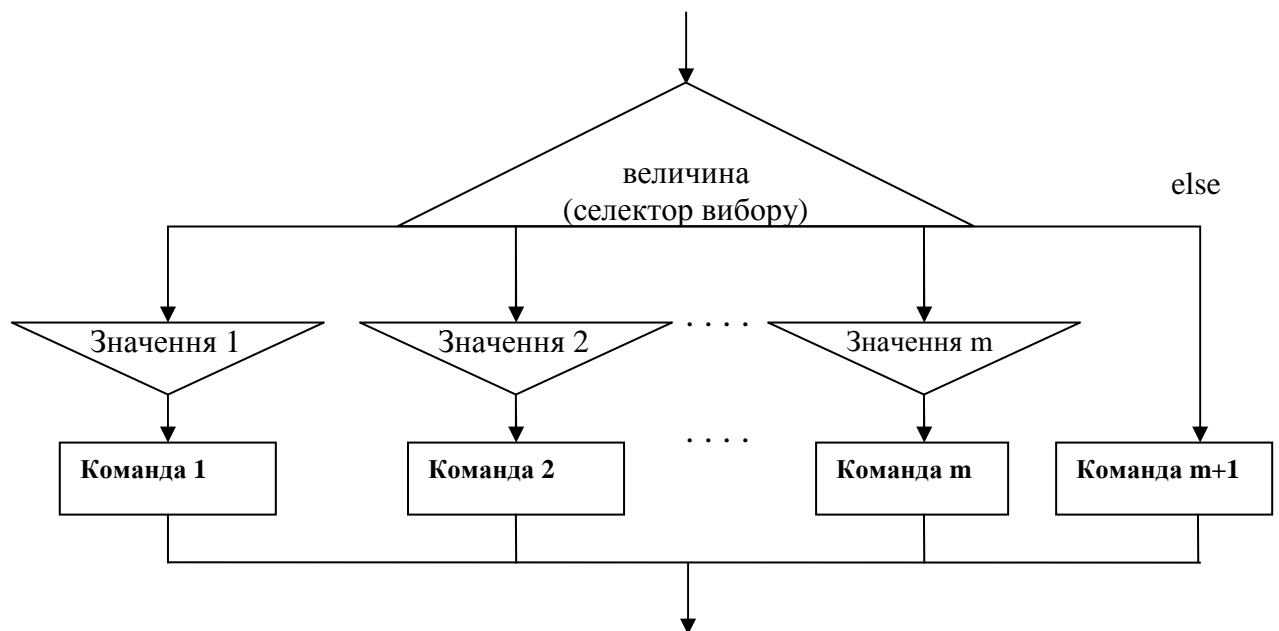


Рис. 7. Блок-схема структури вибору

1.4. Типи алгоритмів

У залежності від структури алгоритму виділяють такі їх типи : лінійні, з розгалуженням та циклічні.

1.4.1.Лінійні алгоритми

Лінійним алгоритмом називається алгоритм, у якому виконуються послідовно всі без виключення команди згідно порядку їх слідування.

Прикладом лінійного алгоритму, записаного з використанням словесного способу опису, може слугувати наступний алгоритм.

Приклад. За допомогою скляних банок на 3л та 5л відміряти 1л води.

Розв'язання

Крок 1. Наповнити 3л банку.

Крок 2. Перелити воду з 3л банки у 5л банку.

Крок 3. Знову наповнити 3л банку.

Крок 4. Доповнити 5л банку водою з 3л. В результаті у 3л банці залишиться 1л води. Кінець алгоритму.

Надалі будемо розглядати тільки графічний спосіб запису алгоритму.

Основою лінійних алгоритмів є структура слідування,

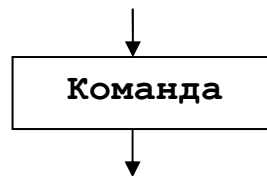


Рис.8

що дозволяє здійснювати перетворення даних без перевірки додаткових умов чи виконання повторюваних операцій.

Приклад. Обчислити площу прямокутника, заданого довжинами його сторін.

Розв'язання

Позначимо сторони прямокутника a і b , а площу S . У даному випадку величини a і b є аргументами, а S – результатом роботи алгоритму.

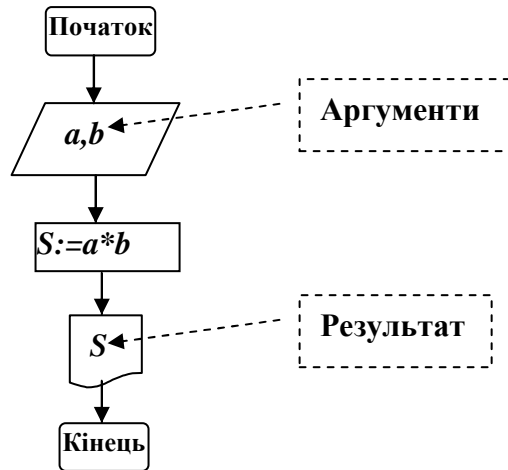


Рис.9

Приклад. Знайти площу трикутника, заданого довжинами своїх сторін.

Розв'язання.

Позначимо сторони трикутника a, b і c , а площу S .

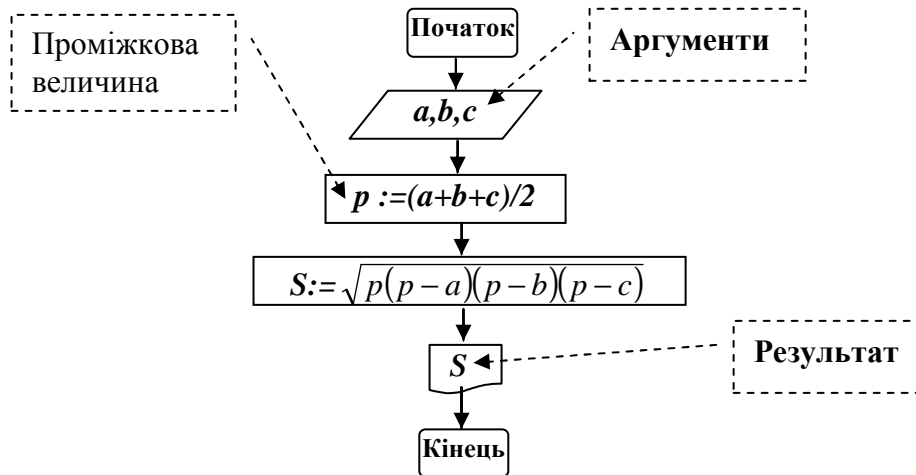


Рис.10

Як бачимо, величина p у даному випадку не є ні аргументом, ні результатом, а лише використовується для збереження проміжкового значення – півпериметру, який використовується при обчисленні площі.

Приклад. Знайти площу фігури, що складається з прямокутника та двох півкіл

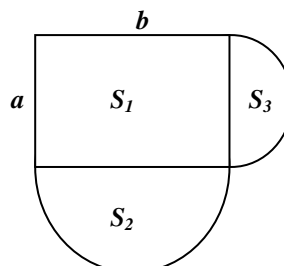


Рис.11

Розв'язання

Площа фігури складається з S_1 – площі прямокутника, S_2 – площа одного півкола, S_3 – площа другого півкола.

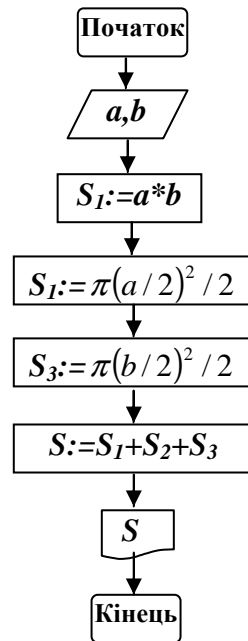


Рис.12

1.4.2. Алгоритми з розгалуженням

Алгоритм з розгалуженням – це алгоритм, що містить хоча б одну умову, в результаті перевірки якої здійснюється перехід до одного з можливих кроків.

Основною алгоритмів з розгалуженням є структури розгалуження, хоча при побудові алгоритмів з розгалуженням можуть бути також використані і структури слідування.

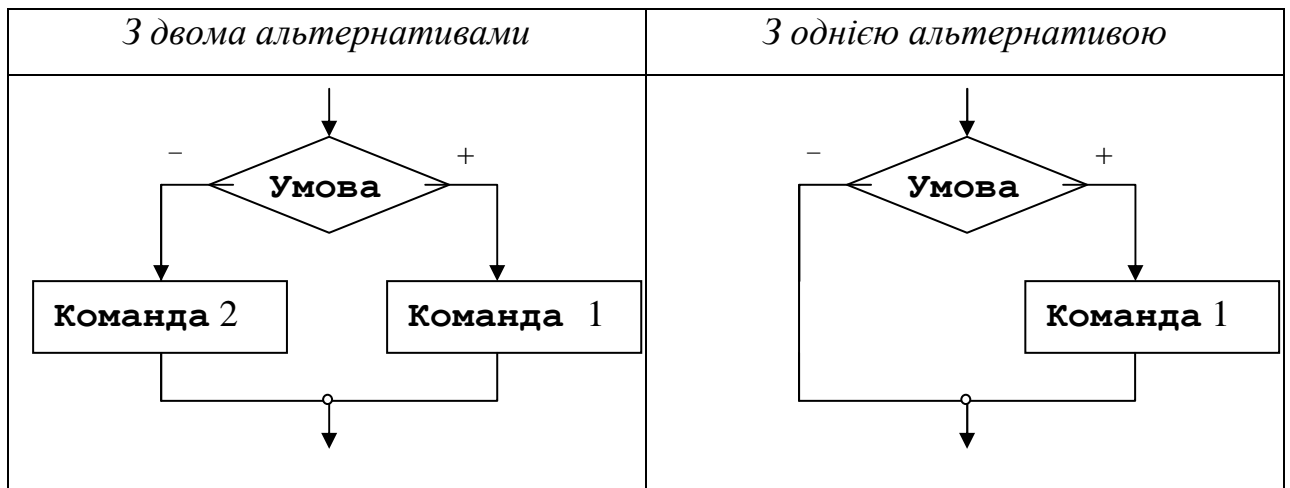


Рис.13. Структури розгалуження

Однак при описі алгоритмів з розгалуженням може використовуватись також і структура вибору

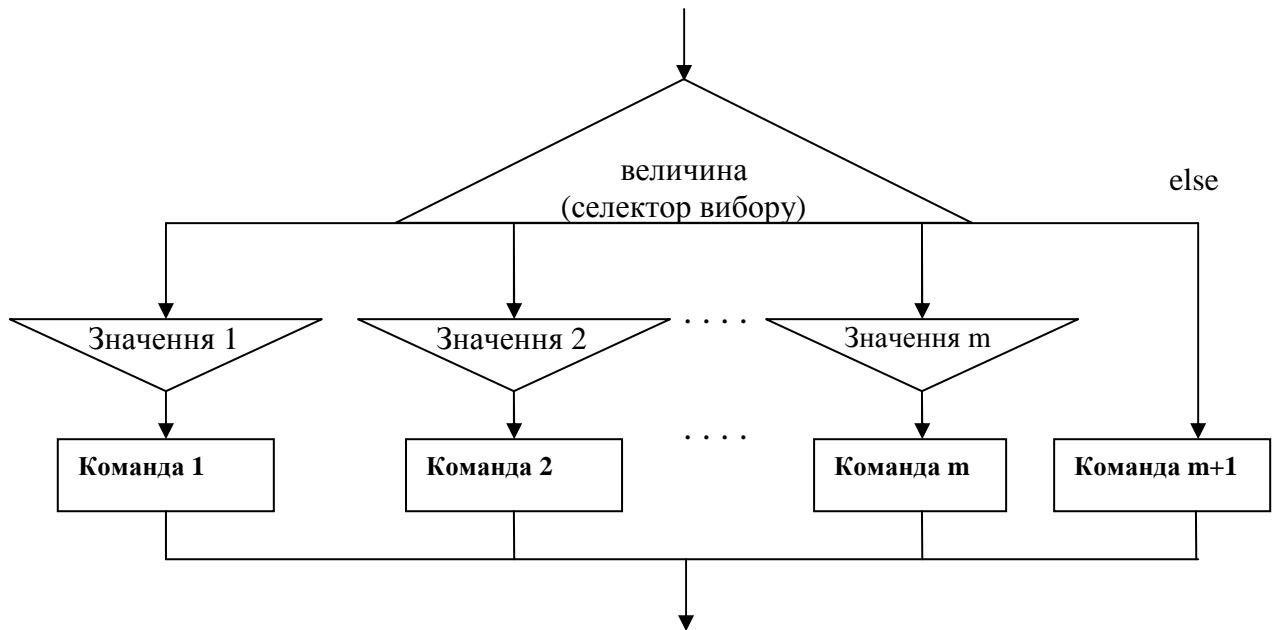


Рис. 14. Структура вибору

Досить часто вибір тих чи інших дій для продовження алгоритму залежить від виконання або невиконання певних умов. Команди, які аналізують ці умови та наступний вибір виконуваних дій алгоритму, називаються командами розгалуження, і є основою структур розгалуження.

Приклад. Обчислити значення виразу, якщо це можливо, або ж вивести повідомлення про неможливість обчислення.

$$S = \frac{2}{n-5}.$$

Розв'язання

Очевидно, обчислити значення виразу неможливо тоді, коли $n - 5 \neq 0$

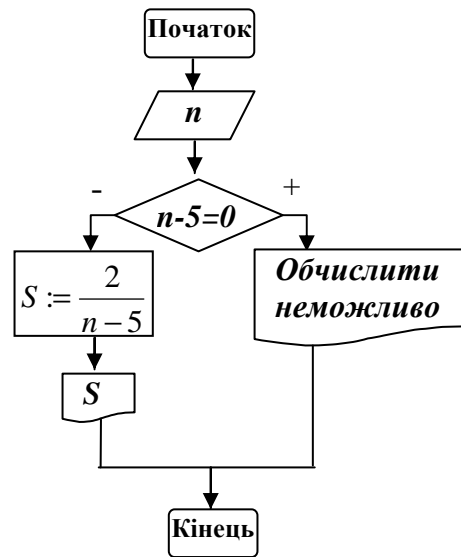


Рис.15

Приклад. Знайти найбільше серед двох чисел.

Розв'язання

Позначимо задані два числа a і b , а максимальне значення max .

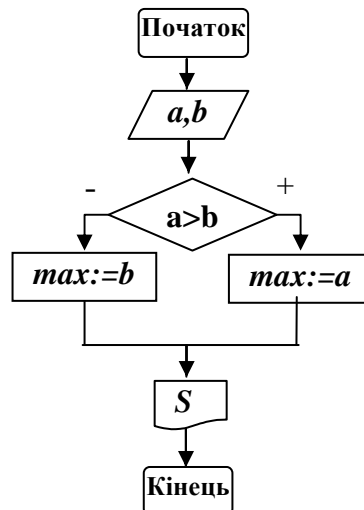


Рис.16

Приклад. Знайти найбільше серед трьох чисел.

Розв'язання

Позначимо задані три числа a, b і c , а максимальне значення max .

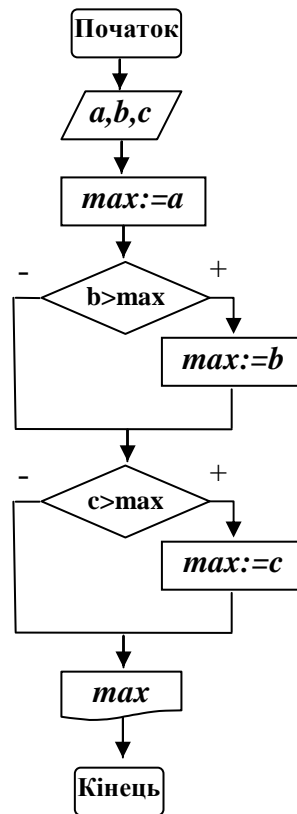


Рис.17

Приклад. Обчислити значення функції

$$f(x) = \begin{cases} 2 + x, & x < 10, \\ x^2, & x \geq 10. \end{cases}$$

Розв'язання

Позначимо $y = f(x)$

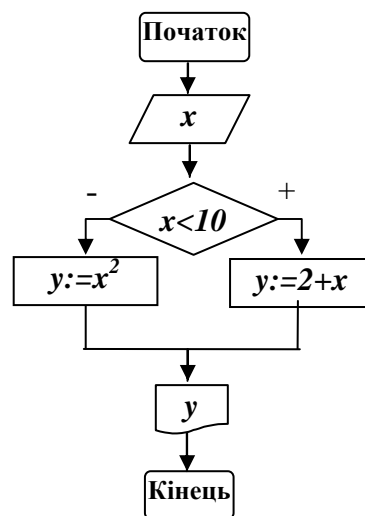


Рис.18

Приклад. Обчислити значення функції

$$f(x) = \begin{cases} \sin x, & x < 2, \\ \sqrt{x}, & 2 \leq x \leq 6, \\ \cos x, & x > 6. \end{cases}$$

Розв'язання

Позначимо $y = f(x)$

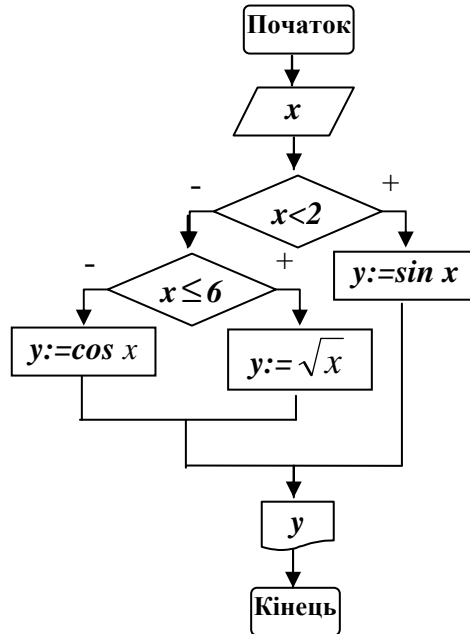


Рис.19

Приклад. З'ясувати, чи є трикутник, заданий довжинами своїх сторін, прямокутним.

Розв'язання

Позначимо сторони трикутника a, b і c .

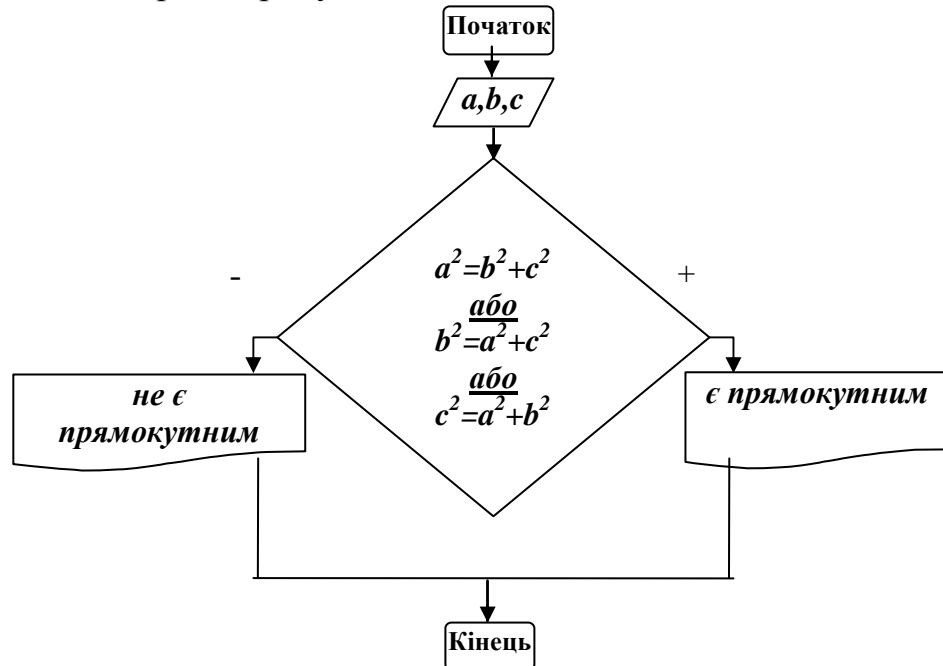


Рис.20

Приклад. Користувач задає оцінку у п'ятибальній шкалі. Необхідно встановити, текстове представлення цієї оцінки (5 – «відмінно», 4 – «добре», 3 – «задовільно», 1 і 2 – «незадовільно»).

Розв'язання

При побудові алгоритму використаємо структуру вибору.

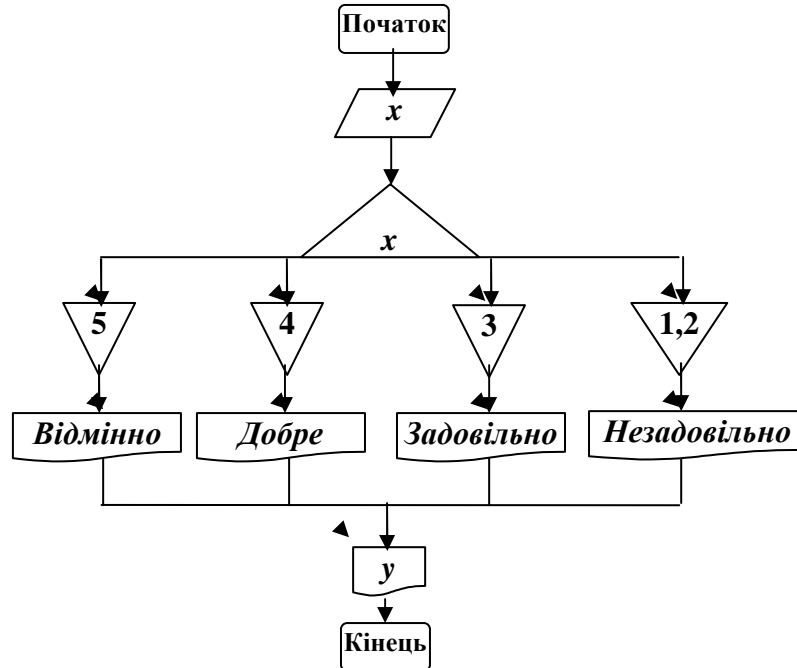


Рис.21

1.4.3. Циклічні алгоритми

Циклічний алгоритм – це алгоритм, у якому певна послідовність команд повторюється кілька разів з новими вхідними даними.

Команди, що дозволяють кілька разів повторювати певний блок команд алгоритму, називаються циклічними. Основною циклічних алгоритмів є структури повторення (структури циклу), серед яких виділяють структури повторення з передумовою, з післяумовою та з параметром.

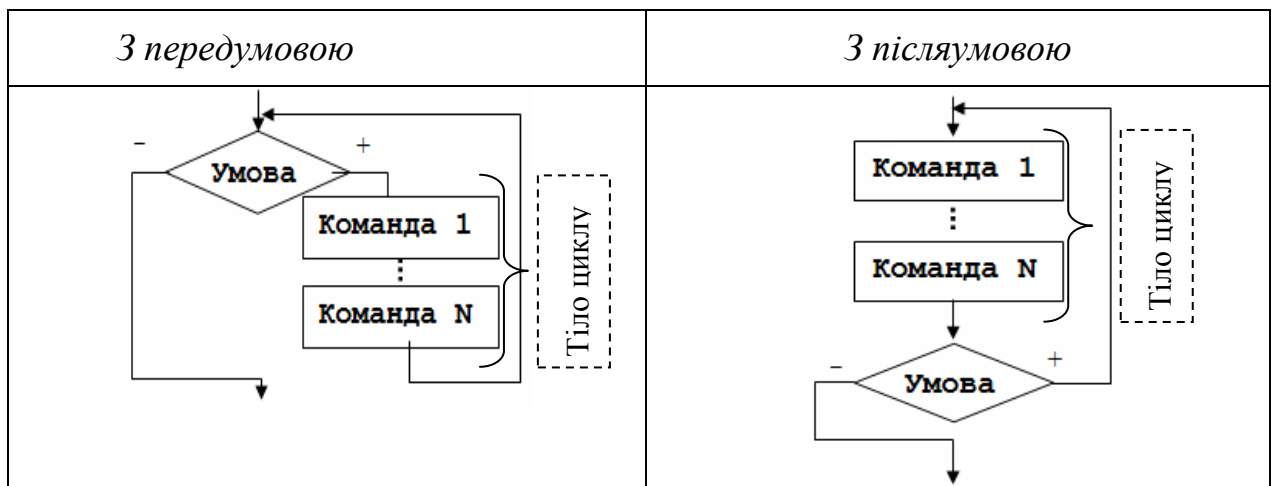


Рис.22

Структура повторення з параметром

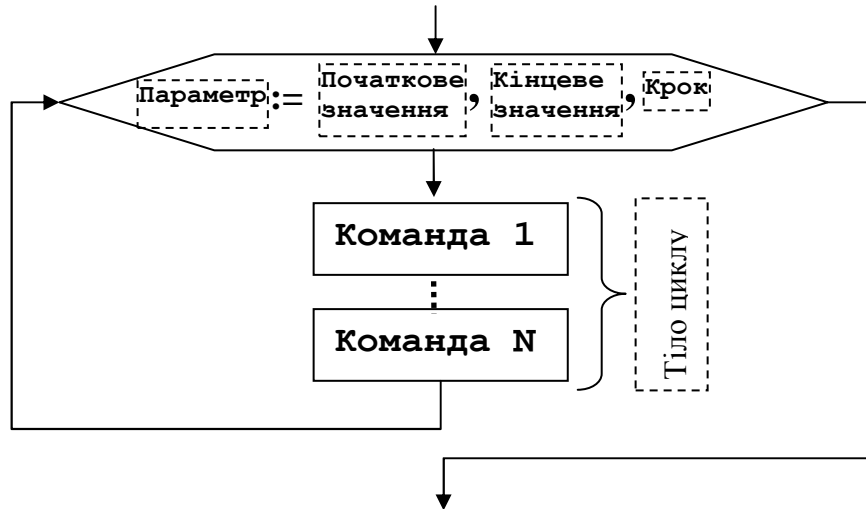


Рис.23

Зауважимо, що у циклічних алгоритмах окрім структур повторення можуть бути використані структури розгалуження та слідування.

Приклад. Обчислити значення виразу

$$S = 2 + 4 + \dots + 214.$$

Розв'язання

У даному виразі необхідно знайти суму доданків, які змінюються. Позначимо доданок, що змінюється a . Його початкове значення 2, додавання необхідно здійснити поки значення доданка менше або рівне 214, сусідні значення доданків змінюються на 2.

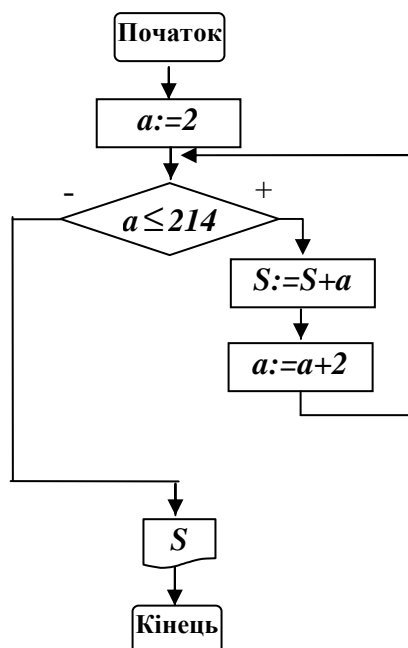


Рис.24

Приклад. Знайти $n!$ *Розв'язання*

Як відомо $n!$ є добутком перших n натуральних чисел

$$n! = 1 * 2 * \dots * n.$$

Тобто необхідно обчислити добуток множників, що змінюються. До того ж відомо початкове значення множника і кінцеве. Тому можемо використати для розв'язання задачі цикл з параметром.

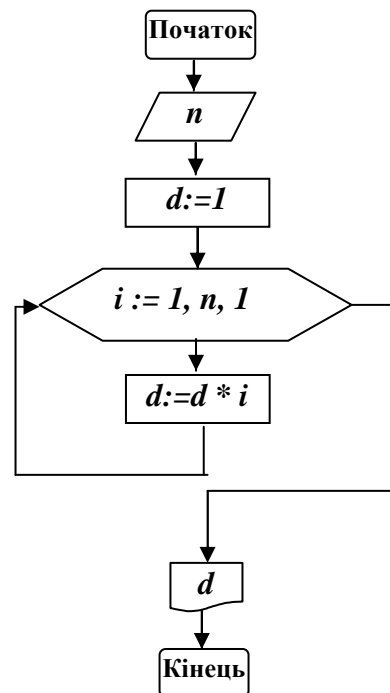


Рис.25

Приклад. Значення поступово задаються користувачем і додаються. Додавання необхідно припинити тоді, коли сума стане більшою за K

Розв'язання

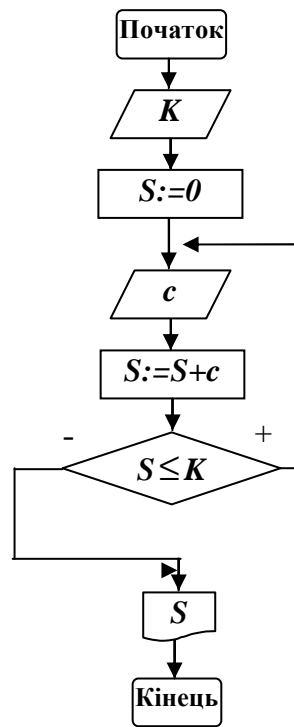


Рис.26

1.5. Структурний підхід до розробки алгоритмів. Допоміжні алгоритми

В міру того, як алгоритми стають дедалі складнішими, відповідно ростуть труднощі з розумінням їх роботи, знаходженні та виправленні помилок, доведенні їх правильності і, при необхідності, правильного внесення в них змін. Вважається, що від 50 до 100% часу програміст витрачає на розв'язання задач по виправленню, експлуатації та модифікації програм. У зв'язку з цим виникає питання більш системного підходу до програмування, тобто розробка методик, використання яких зменшить ймовірність помилок в програмах, спростить їх розуміння і полегшить модифікацію.

На сьогоднішній день широке застосування отримала методика *структурного програмування зверху-вниз*. Суть структурного підходу визначається такими засадами:

1) складна задача розбивається на простіші, що легко сприймаються, частини, кожна з яких має один вхід і один вихід;

2) логіка побудови алгоритму повинна спиратися на мінімальне число базисних структур алгоритмів (слідування, розгалуження, циклу).

Структурний підхід до побудови алгоритмів і програм базується на теоремі про структурування, доведеної у 1965 р. італійськими вченими Бомом і Джакопіні. Згідно з цією теоремою, будь-який правильний алгоритм можна подати за допомогою тільки трьох базисних структур алгоритмів: слідування, розгалуження і повторення (циклу). Алгоритм правильний, якщо він має один вхід та один вихід, не містить нескінченних циклів і невиконуваних вказівок (команд). Будь-який правильний алгоритм, згідно з цією теоремою, можна розбити на окремі блоки. Кожен із цих блоків перетворюється в одну з трьох базових структур або їх комбінацію, у результаті чого створюється алгоритм, функціонально еквівалентний початковому і складений тільки з *базисних структур алгоритмів*.

При застосуванні структурного підходу алгоритми формуються спочатку у «найзагальніших» вказівках, при цьому в записі алгоритму можуть використовуватися вказівки, що виходять за межі можливостей виконавця. Тоді на кожному наступному етапі окремі деталі алгоритму уточнюються. Процес продовжується до тих пір, поки всі алгоритми не будуть складатися із вказівок, зрозумілих виконавцю.

Приклад. Обчислити значення виразу

$$S = \max\{\min\{a, b\}, \max\{-a, b\}\} + \max\{a, -b\}.$$

Розв'язання

Введемо позначення для деяких величин, які необхідно обчислити в процесі розв'язання задачі

$$S = \max\left\{ \underbrace{\left\{ \underbrace{\min\{a, b\}}_{d_1}, \underbrace{\max\{-a, b\}}_{d_2} \right\}}_{m_1} + \underbrace{\max\{a, -b\}}_{m_2} \right\}.$$

Для побудови алгоритму розв'язання даної задачі скористаємось структурним підходом.

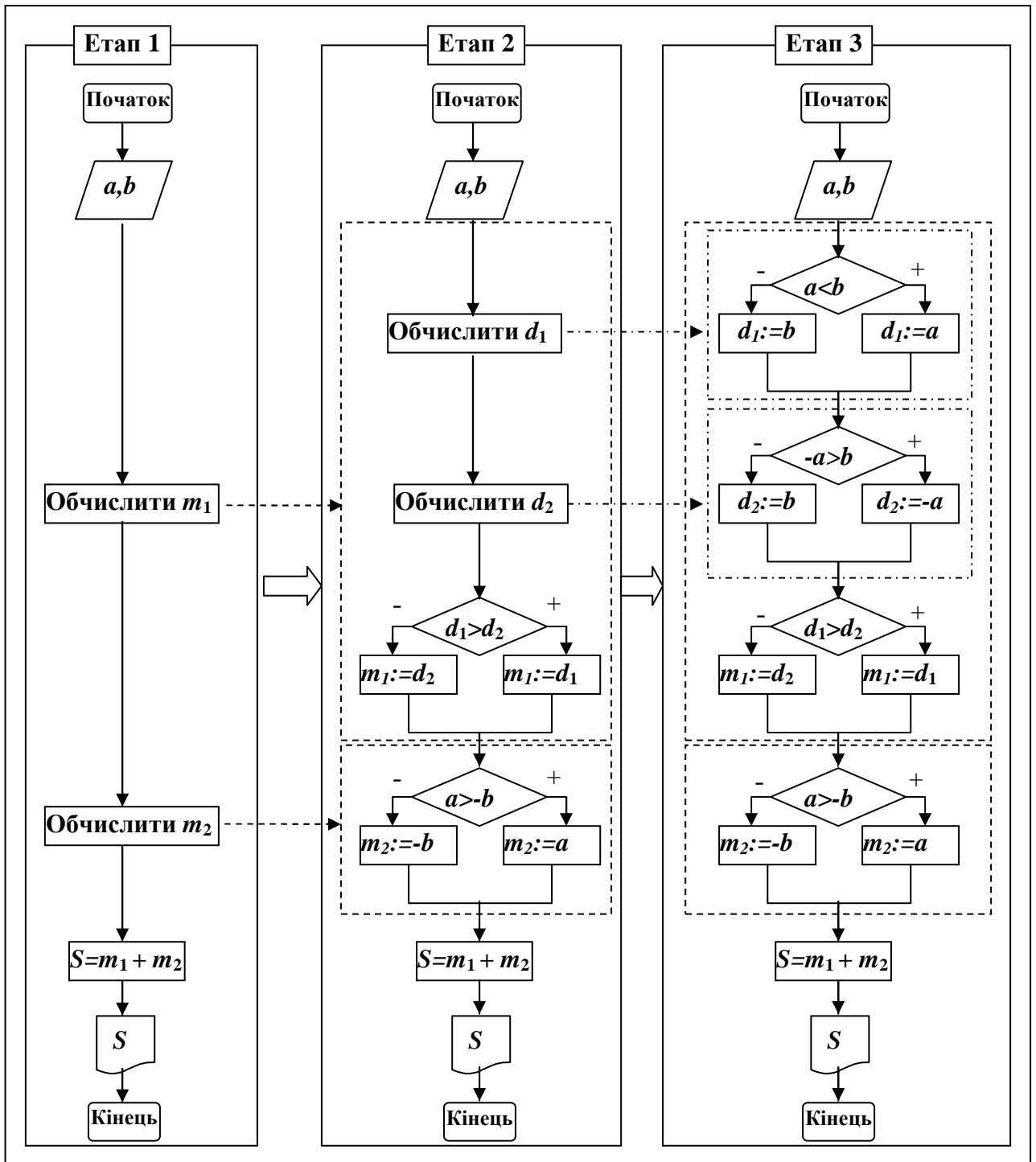


Рис.27. Застосування структурного підходу для обчислення

значення виразу

$$S = \max \left\{ \underbrace{\min\{a, b\}}_{d_1}, \underbrace{\max\{-a, b\}}_{d_2} \right\} + \underbrace{\max\{a, -b\}}_{m_2}$$

m_1

Як бачимо, у даному випадку, побудову алгоритму здійснено за допомогою трьох етапів. При цьому на кожному наступному етапі деталізується

виконання деяких команд, що описані з використанням узагальнених вказівок. В результаті на третьому етапі одержано алгоритм, що не містить «узагальнених» вказівок і описаний з використанням тільки базових структур.

Часто при розробці алгоритмів доводиться описувати послідовності операцій, що розв'язуються одну й ту ж підзадачу, але для різних величин. Так, наприклад, у попередньому прикладі нам довелося тричі знаходити найбільше серед двох чисел ($\max\{-a, b\}$, $\max\{a, -b\}$, $\max\{d_1, d_2\}$). Також виникають питання повторного використання розроблених раніше алгоритмів, колективної розробки алгоритмів декількома програмістами та можливості розбиття складної задачі на підзадачі і окремого їх розв'язання. Для подолання цих труднощів було введено поняття допоміжного алгоритму. *Допоміжним алгоритмом* називають алгоритм, який створений наперед і викликається та повністю виконується всередині іншого, *основного алгоритму*.

Опис допоміжних алгоритмів

При описі допоміжного алгоритму дотримуємось загальних правил опису алгоритмів з певними відмінностями у позначеннях початку/кінця алгоритму та виведенні (поверненні) результатів.

Опис початку і кінця допоміжного алгоритму

При описі функціонального блоку, що позначає початок допоміжного алгоритму необхідно вказати назву допоміжного алгоритму та список формальних параметрів. *Формальні параметри* – це величини, які необхідні для роботи допоміжного алгоритму та величини, що є результатами роботи допоміжного алгоритму.

При описі функціонального блоку, що позначає кінець допоміжного алгоритму вказують не тільки службове слово «**Кінець**» а й ім'я допоміжного алгоритму.

Таблиця 1. Опис початку/кінця допоміжного алгоритму

Загальна форма	Приклад
<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Назва алгоритму (Список формальних параметрів) </div> <div style="text-align: center;"> ↓ ⋮ ↓ </div> <div style="border: 1px solid black; padding: 5px; margin-left: auto; margin-right: auto;"> Кінець Назва алгоритму </div>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <i>Max</i>(<i>c</i>₁, <i>c</i>₂) </div> <div style="text-align: center;"> ↓ ⋮ ↓ </div> <div style="border: 1px solid black; padding: 5px; margin-left: auto; margin-right: auto;"> Кінець <i>Max</i> </div>

Передача даних у допоміжні алгоритми під час їх виклику

Виклик допоміжного алгоритму виконується за допомогою функціонального блоку



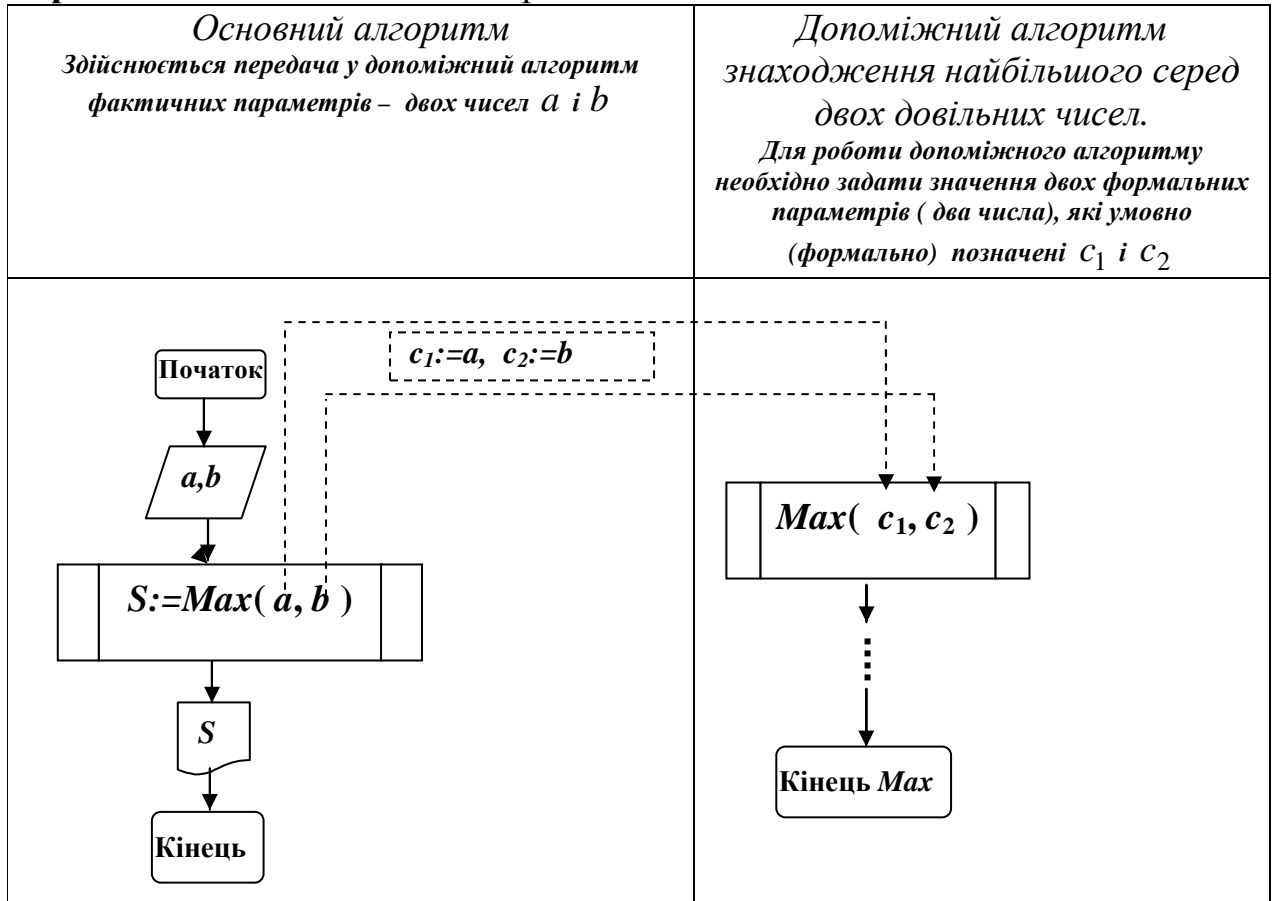
Рис.28

При цьому допоміжному алгоритму передаються вхідні дані (*фактичні параметри*), які необхідні для виконання допоміжного алгоритму, а допоміжний алгоритм може передавати в основний алгоритм деякі вихідні значення (*результати*) та змінювати значення вхідних величин.

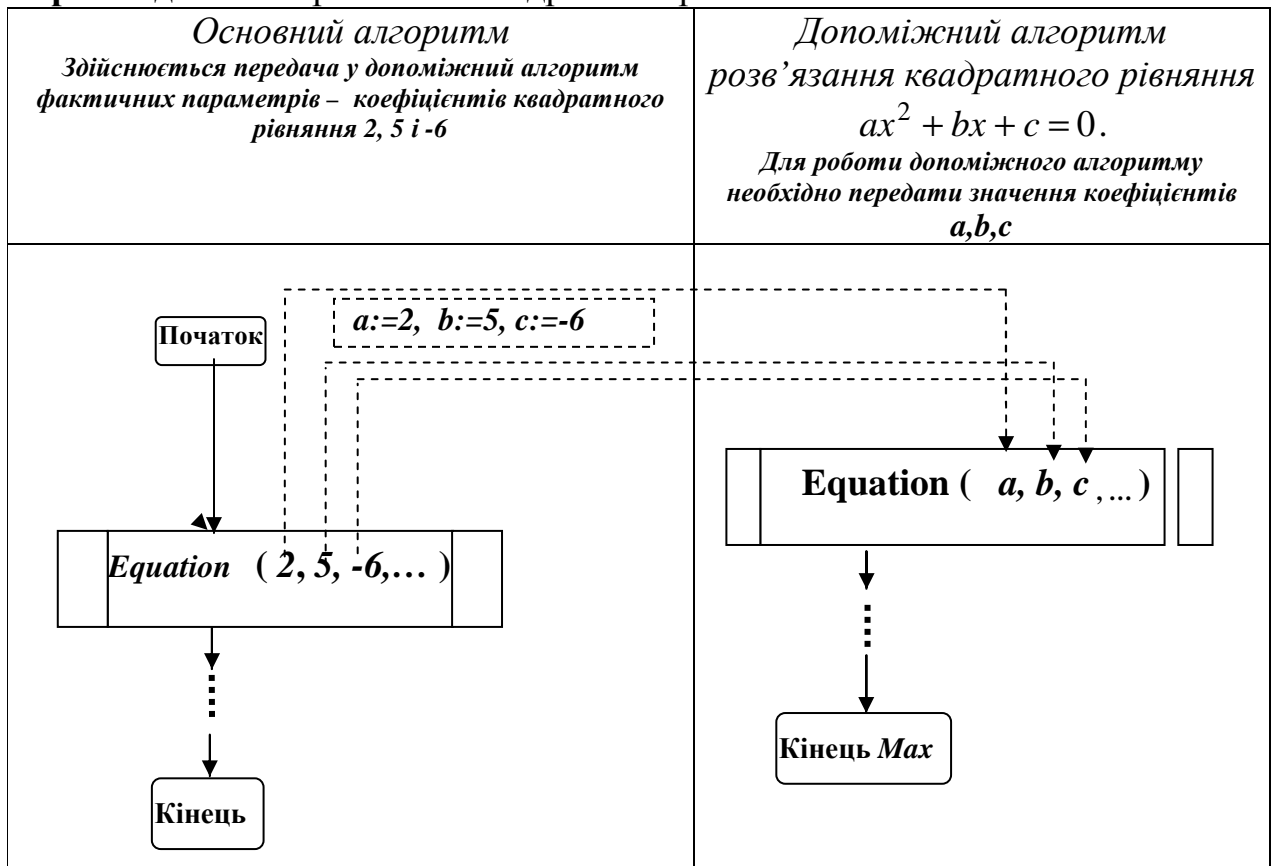
Формальні параметри одержують значення фактичних у порядку їх слідування: перший формальний параметр одержує значення першого фактичного параметра, другий формальний параметр – другого фактичного параметра і т.д.

Наведемо декілька прикладів передачі даних у допоміжні алгоритми

Приклад. Знайти найбільше серед двох чисел a і b .



Приклад. Знайти розв'язки квадратного рівняння $2x^2 + 5x - 6 = 0$.



Повернення результатів

Результати роботи допоміжного алгоритму можуть повертатися у основний алгоритм через формальні параметри або ж ім'я алгоритму.

Повернення результату через формальні параметри

Як було зазначено раніше серед формальних параметрів можуть бути не тільки величини, які необхідні для роботи алгоритму, а й величини, що є результатами роботи алгоритму. При цьому вказуючи фактичні параметри, необхідно вказати величини, у яких буде збережено результат.

Приклад. Знайти значення величини d , якщо

$$d = \max\{a, b\}.$$

Розв'язання

Для розв'язання даної задачі використаємо допоміжний алгоритм, який знаходить найбільше із двох значень c_1 і c_2 та зберігає його у змінній m .

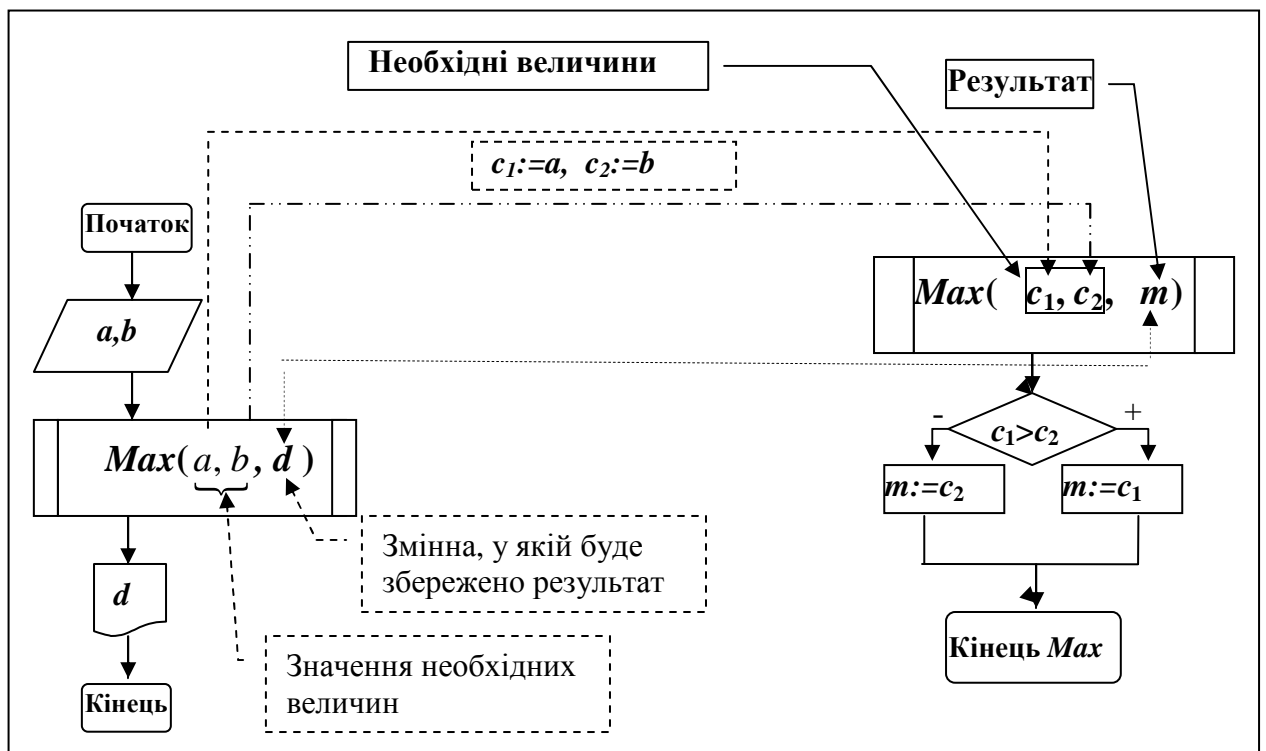


Рис.29

Приклад. Знайти площу та периметр трикутника, заданого довжинами своїх сторін: 2, 4, 3.

Розв'язання

Для розв'язання даної задачі використаємо допоміжний алгоритм *GetSP*, що дозволяє знаходити S – площу та P – периметр трикутника, заданого довжинами сторін: a, b, c .

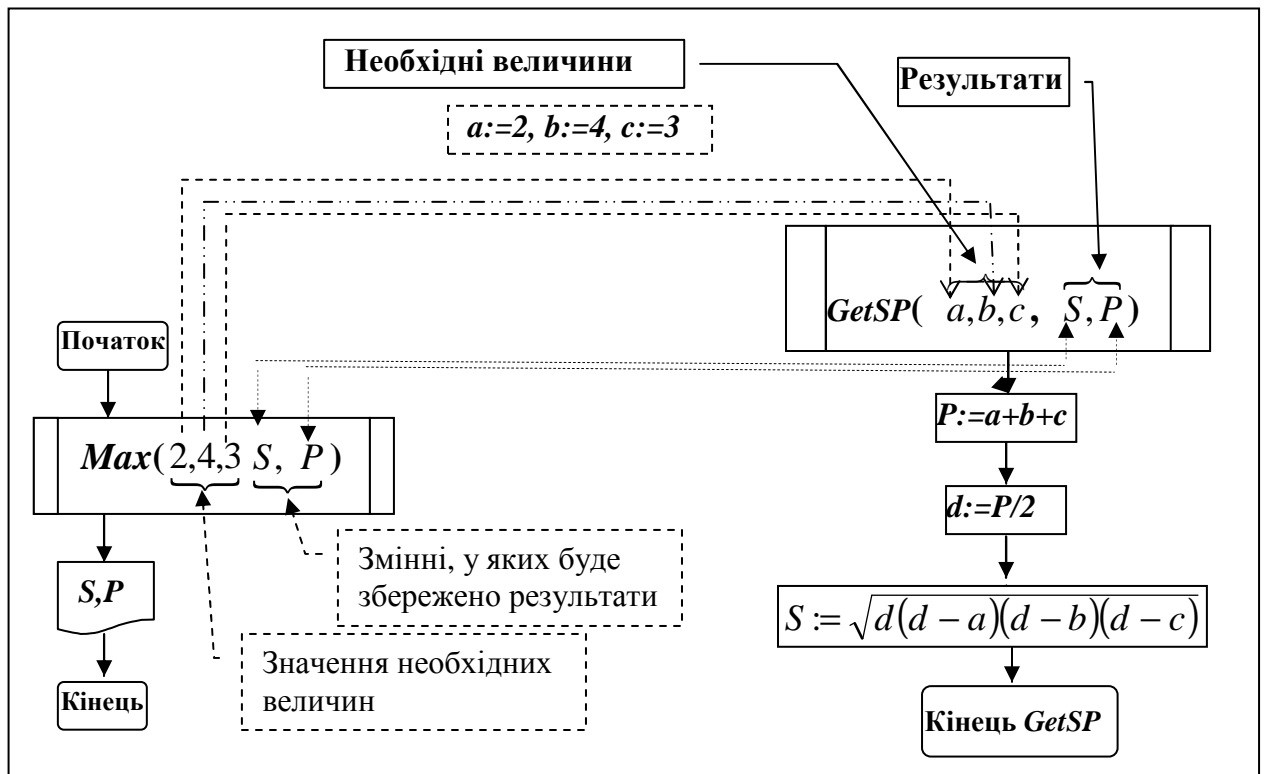


Рис.30

Бувають також випадки, коли величини, необхідні для роботи допоміжного алгоритму є одночасно і результатами роботи цього алгоритму.

Приклад. Більше з двох різних чисел x і y необхідно помножити на 2, а менше з них – поділити на 3.

Розв'язання

Для розв'язання цієї задачі використаємо допоміжний алгоритм *Change*, який дозволяє виконати необхідні операції над деякими двома числами, що умовно позначено a і b . У даному випадку величини a і b є як величинами, що необхідні для роботи допоміжного алгоритму, так і величинами, де зберігаються результати, оскільки метою алгоритму є зміна значень цих величин.

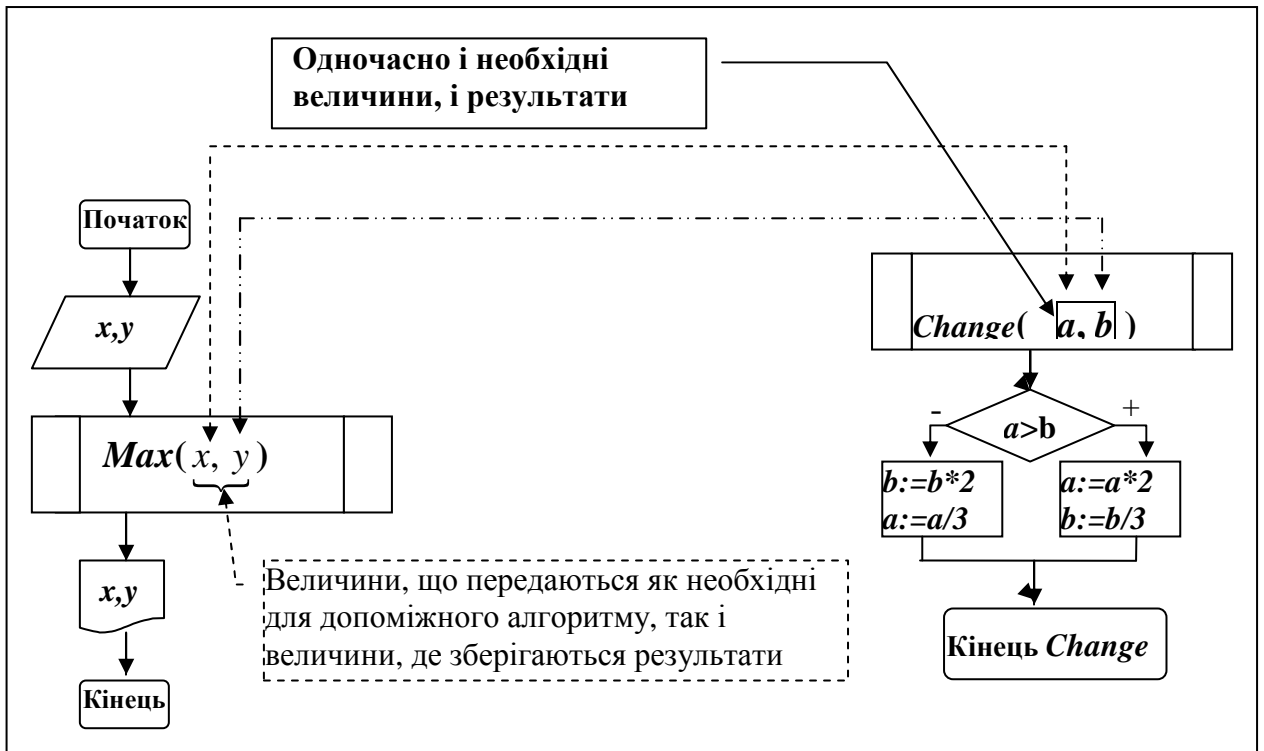


Рис.31

Повернення результату через ім'я допоміжного алгоритму

Повернення однієї величини, як результату, через ім'я допоміжного алгоритму може бути здійснене з використанням службового слова **return**.

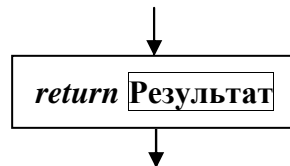


Рис. 32

Приклад. Обчислити значення виразу

$$S = \max\{a, b\} + \max\{b, c\}.$$

Розв'язання

Позначимо проміжкові величини, що необхідно обчислити при розв'язанні задачі

$$S = \underbrace{\max\{a, b\}}_{d_1} + \underbrace{\max\{b, c\}}_{d_2}.$$

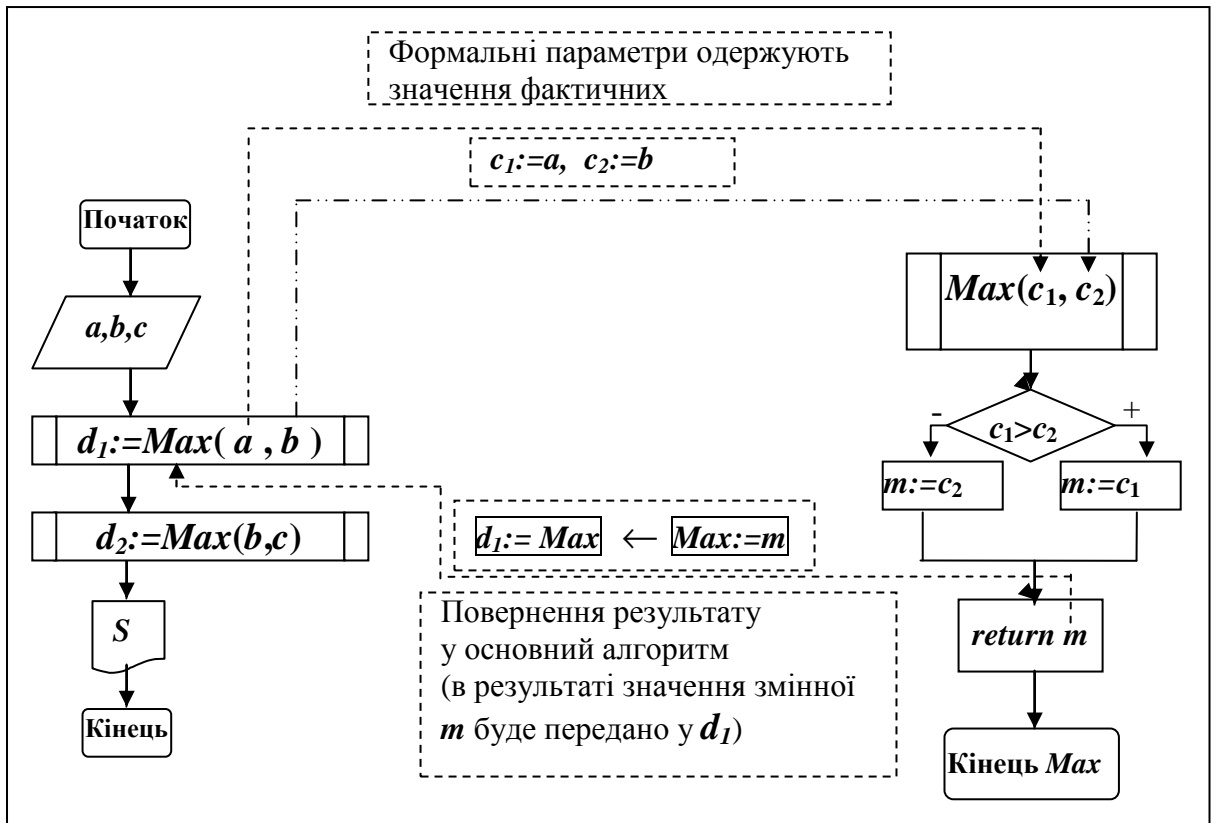


Рис.33. Обчислення значення виразу $S = \underbrace{\max\{a, b\}}_{d_1} + \underbrace{\max\{b, c\}}_{d_2}$

Зауважимо, що можливість повернення одного значення через ім'я допоміжного алгоритму дає можливість не використовувати проміжкових величин (як у попередньому прикладі d_1 та d_2), а одразу використовувати результати роботи допоміжних алгоритмів при обчисленні інших виразів. Так попередню задачу можна розв'язати наступним чином.

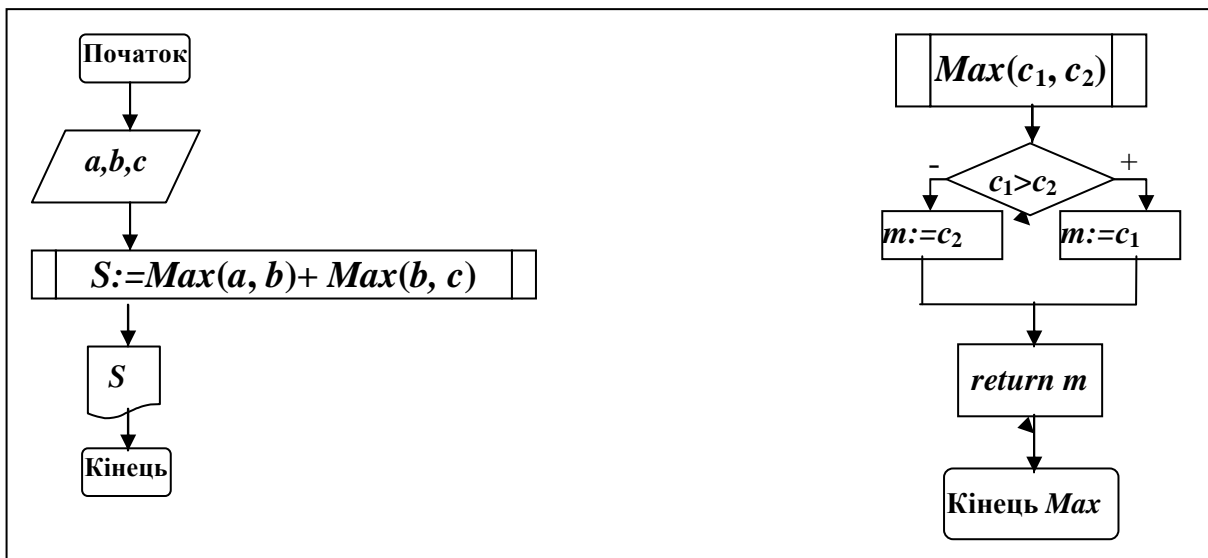


Рис.34

При розробці алгоритмів можна використовувати обидва способи повернення результатів, як через формальні параметри, так і через ім'я допоміжного алгоритму.

Приклад. Знайти розв'язок квадратного рівняння $2x^2 + 5x - 6 = 0$.

Розв'язання

Для розв'язання даної задачі використаємо допоміжний алгоритм *Equation*, який дозволяє знаходити розв'язки довільного квадратного рівняння $ax^2 + bx + c = 0$ і повертати розв'язки через формальні параметри x_1 і x_2 . Також через ім'я допоміжного алгоритму повертається кількість розв'язків рівняння (0 – немає розв'язків, 1 – розв'язок тільки один, 2 – розв'язків два).

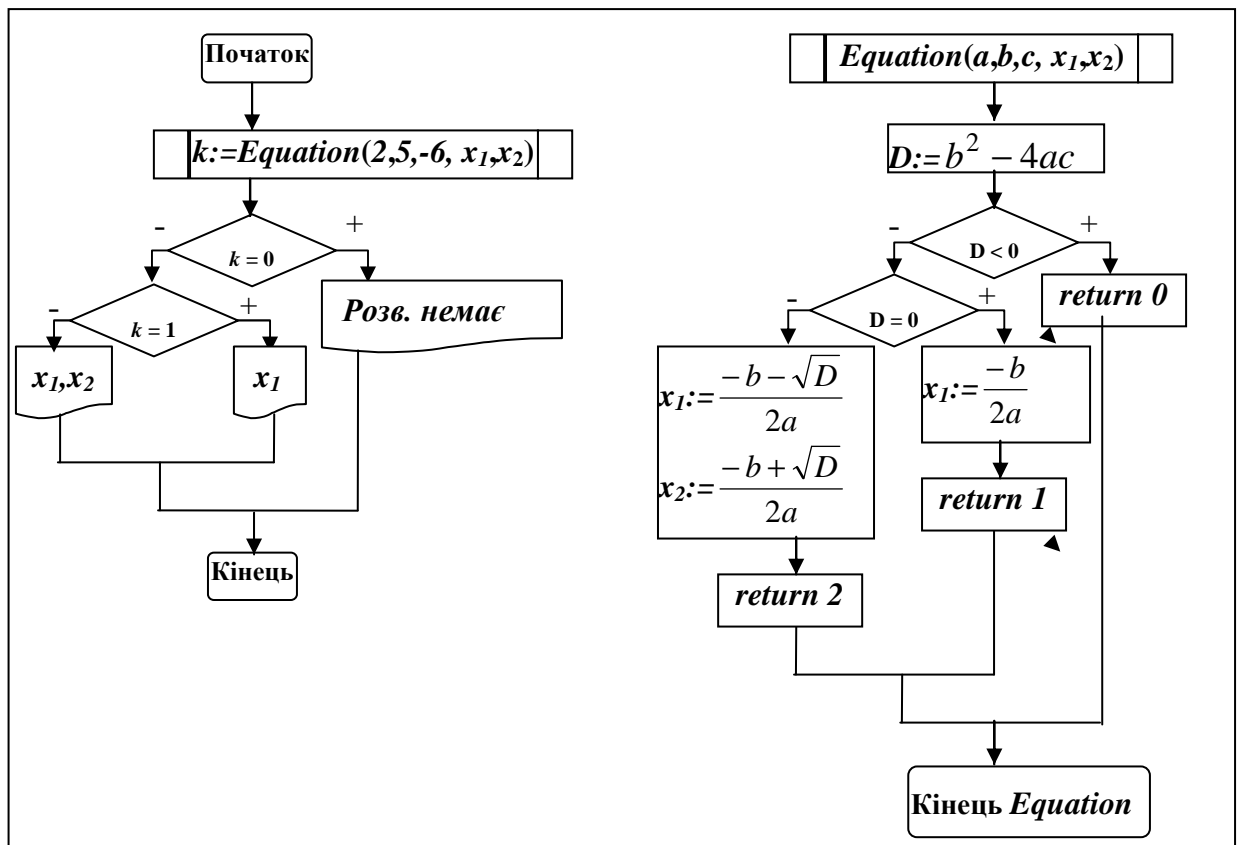


Рис.35

Структурний підхід може бути реалізовано з використанням допоміжних алгоритмів. При цьому складна задача розбивається на більш прості, що легко сприймаються підзадачі. Розв'язання кожної із підзадач реалізовується у вигляді допоміжних алгоритмів. Це дає можливість сконцентрувати

зусилля на розв'язуванні кожної підзадачі окремо, що в свою чергу значно спрощує процес розробки.

Рекурсивні алгоритми

Рекурсивний алгоритм – це алгоритми, який в якості допоміжного використовує сам себе. Іншими словами, у рекурсивному алгоритмі міститься виклик цього ж алгоритму, але з іншими вхідними даними.

Такого роду алгоритми виникають у випадку, коли для розв'язання деякої підзадачі необхідно розв'язати іншу однотипну підзадачу з іншими вхідними даними.

Приклад. Обчислити $n!$.

Розв'язання

Згідно з властивостями факторіалів можемо записати

$$n! = \begin{cases} 1, & n = 1, \\ n * (n - 1)!, & \text{інакше.} \end{cases}$$

Тобто для знаходження значення $n!$ необхідно розв'язати однотипну задачу знаходження факторіалу $(n - 1)!$. А тому, дана задача може бути розв'язана з використанням рекурсивного допоміжного алгоритму.

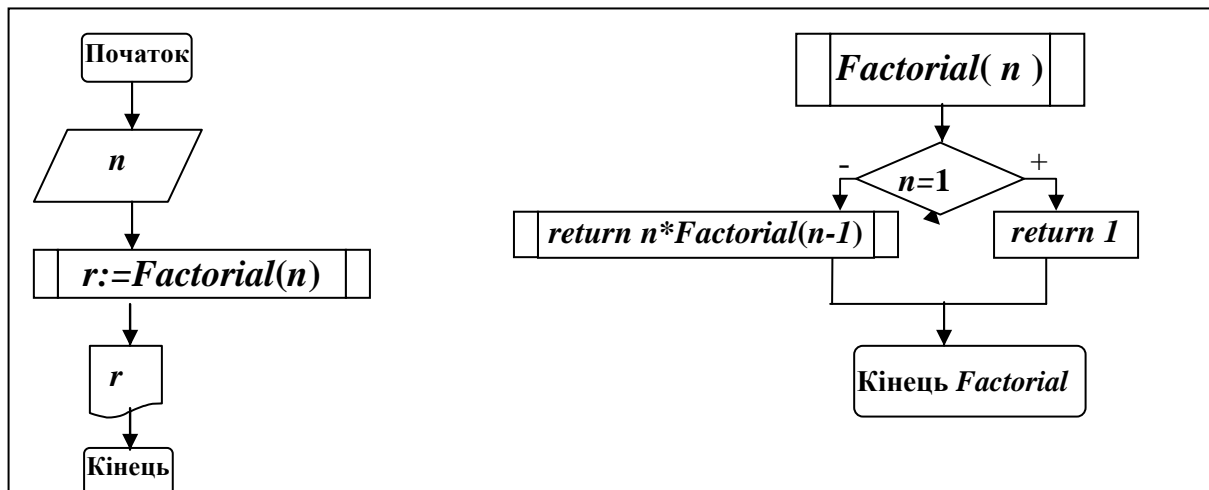


Рис.36

Приклад. Знайти n -тє число Фібоначчі.

Розв'язання

Як відомо, i -ве число Фібоначчі f_i може бути знайдене згідно з умови

$$f_i = \begin{cases} 1, & \text{якщо } i=1 \text{ або } i=2, \\ f_{i-1} + f_{i-2}, & i \geq 3. \end{cases}$$

Знову ж таки, для знаходження i -го ($i \geq 3$) числа Фібоначчі f_i нам спочатку необхідно знайти два попередніх числа f_{i-1} та f_{i-2} .

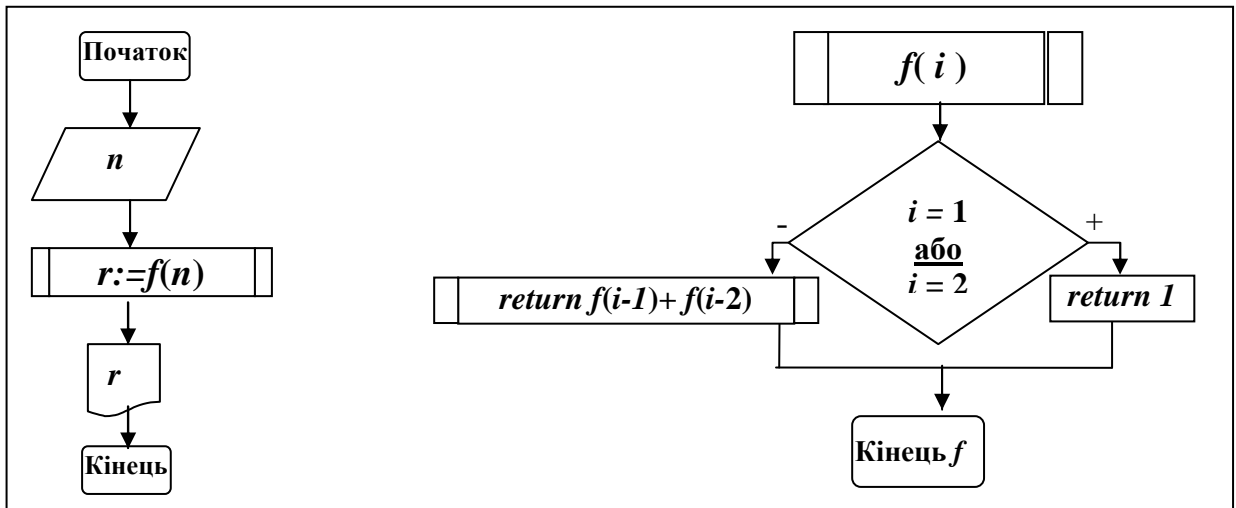


Рис.37

2. РОЗРОБКИ ТЕСТІВ

2.1. Основні поняття та означення

1. Завершіть твердження. Мухаммед бен-муса аль-Хорезмі ...
 - 1) дав чітке означення алгоритму;
 - 2) описав способи запису алгоритмів;
 - 3) дав опис правил виконання дій над десятковими числами;
 - 3) немає правильної відповіді.
2. Завершіть означення. *Алгоритм* – це ...
 - 1) множина інструкцій, з яких виконавець може вибирати ті, які у даний момент необхідно виконати;
 - 2) послідовність чітких інструкцій або команд, які необхідно виконати для розв'язування певної задачі;
 - 3) немає правильної відповіді.
3. Яка з властивостей не є характерною властивістю алгоритмів?
 - 1) Скінченність алгоритмів.
 - 2) Визначеність вказівок алгоритмів.
 - 3) Дискретність вказівок алгоритмів.
 - 4) Всі перераховані властивості є характерними властивостями алгоритмів.
4. Завершіть означення. Властивість *визначеності вказівок алгоритмів* полягає у тому, що ...
 - 1) виконавець сам визначає дії, що необхідно виконати на кожному кроці;
 - 2) кожен крок визначається однозначно і не допускає довільного трактування;
 - 3) немає правильної відповіді.
5. Завершіть означення. Властивість *дискретності вказівок алгоритмів* полягає у тому, що ...
 - 1) для кожного з кроків чітко визначається момент початку та завершення;
 - 2) кожен крок визначається однозначно і не допускає довільного трактування;
 - 3) немає правильної відповіді.
6. Завершіть означення. Властивість *скінченності алгоритмів* полягає у тому, що ...
 - 1) для кожного з кроків виконавець чітко може визначити момент завершення одного кроку і початок іншого;
 - 2) результат одержуємо за скінченну кількість кроків;
 - 3) кожен крок визначається однозначно і не допускає довільного трактування.
7. Завершіть означення або зазначте, що правильної відповіді немає.

Властивість *масовості* полягає у тому, що ...

- 1) алгоритм може бути використаним для різних вхідних даних;
- 2) результат одержуємо за скінчену кількість кроків;
- 3) виконавець може виконувати алгоритм декілька разів;
- 4) немає правильної відповіді.

8. Завершіть означення або зазначте, що правильної відповіді немає.

Властивість *зрозумілості* полягає у тому, що ...

- 1) алгоритм має складатися з команд, які входять до системи команд його виконавця (зрозумілі для виконавця);
- 2) виконавець може пропускати команди, які для нього є незрозумілими;
- 3) немає правильної відповіді.

9. Завершіть означення або зазначте, що правильної відповіді немає.

Властивість *результативності алгоритмів* полягає у тому, що ...

- 1) за скінченну кількість кроків алгоритм має приводити до одержання результату або ж до досягнення поставленої мети;
- 2) кожна команда алгоритму однозначно встановлює, що треба робити на даному кроці;
- 3) немає правильної відповіді.

10. Які алгоритми називають *еквівалентними*?

- 1) Алгоритми, які при однакових вхідних даних призводять до одержання однакових результатів.
- 2) Алгоритми, які може виконувати один і той же виконавець.
- 3) Немає правильної відповіді.

11. Завершіть означення або зазначте, що правильної відповіді немає.

Система команд виконавця – це ...

- 1) це точний опис дій та послідовності їх виконання (відсутність їх неоднозначного тлумачення);
- 2) послідовність команд, які йому необхідно виконати;
- 3) множина команд, які він може виконати.

12. Завершіть означення або зазначте, що правильної відповіді немає.

Складність алгоритму – це ...

- 1) кількісна характеристика, яка визначається часом, за який виконується алгоритм, та об'ємом пам'яті комп'ютера, необхідним для реалізації цього алгоритму;
- 2) кількісна характеристика, яка визначається часом, за який було складено алгоритм;
- 3) немає правильної відповіді.

13. Що визначає часова складність алгоритму?

- 1) Це час, який необхідний для побудови алгоритму.
- 2) Це час, який необхідний для складання структури алгоритму.
- 3) Це час, який необхідний для аналізу результатів.
- 4) Немає правильної відповіді.

14. Що визначає ємнісна складність алгоритму?

- 1) Об'єм пам'яті комп'ютера, необхідним для виконання цього алгоритму.
- 2) Об'єм пам'яті на диску, що необхідний для збереження цього алгоритму;
- 3) Кількість величин, що розглядаються у даному алгоритмі.
- 4) Немає правильної відповіді.
15. Завершіть означення або зазначте, що правильної відповіді немає. Стосовно алгоритмів *аргументи* – це ...
- 1) поняття, що аргументують правила застосування алгоритму;
- 2) аргументовані пояснення кожного з кроків алгоритму;
- 3) вхідні дані, що необхідні для виконання алгоритму;
- 4) немає правильної відповіді.
16. Завершіть означення або зазначте, що правильної відповіді немає. Стосовно алгоритмів *результати* – це ...
- 1) дані, що одержуємо на кожному із кроків;
- 2) дані, що одержуємо як результат виконання алгоритму;
- 3) дані, що необхідні для виконання алгоритму;
- 4) немає правильної відповіді.
17. Завершіть означення або зазначте, що правильної відповіді немає. Стосовно алгоритмів *проміжкові дані* – це ...
- 1) дані, що одержуємо як результат виконання алгоритму;
- 2) дані, що не є ні аргументами, ні результатами;
- 3) немає правильної відповіді.
18. Завершіть твердження. Виконавцем алгоритму може бути
- 1) тільки людина;
- 2) тільки обчислювальна машина;
- 3) немає правильної відповіді.
19. Завершіть твердження. При виконанні алгоритму виконавець ...
- 1) може довільно трактувати кожен крок алгоритму;
- 2) може змінювати послідовність виконання кроків довільним чином;
- 3) немає правильної відповіді.
20. Яке із тверджень має місце?
- 1) Для розв'язання всіх задач розроблено відповідні алгоритми їх розв'язання.
- 2) Для розв'язання кожної із задач існує єдиний алгоритм її розв'язання.
- 3) Немає правильної відповіді.
21. Чи вірним є твердження?

Кількість алгоритмів, що дозволяють розв'язати деяку задачу залежить від способу запису.

- 1) Тільки для деяких задач.
- 2) Твердження невірне.
- 3) Твердження є вірним для усіх задач.
- 4) Немає правильної відповіді.

2.2. Способи запису алгоритмів

1. Який із способів не може бути використаний для запису алгоритму?
 - 1) За допомогою блок-схем.
 - 2) За допомогою мови програмування C#.
 - 3) За допомогою розмовної мови.
 - 4) Немає правильної відповіді.
2. Завершіть твердження. Спосіб запису алгоритмів залежить від ...
 - 1) системи команд виконавця;
 - 2) від аргументів;
 - 3) від результатів;
 - 4) Немає правильної відповіді.
3. У якому із способів запису алгоритму може бути використано службове слово «якщо»?
 - 1) Графічний спосіб.
 - 2) За допомогою псевдокодів.
 - 3) За допомогою мови програмування Pascal.

4. Фрагмент якого способу запису алгоритму наведено нижче?

Крок 2. Порівняємо два числа і виберемо найбільше.

- 1) Графічний спосіб.
 - 2) За допомогою псевдокодів.
 - 3) За допомогою мови програмування Pascal.
 - 4) Немає правильної відповіді.
5. Фрагмент якого способу запису алгоритму наведено нижче?

**ЯКЩО $n > m$
ТО $n := n - m$
ІНАКШЕ $m := m - n$**

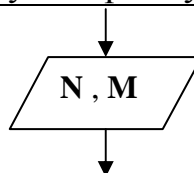
- 1) Графічний спосіб.
 - 2) За допомогою псевдокодів.
 - 3) За допомогою мови програмування Pascal.
 - 4) Немає правильної відповіді.
6. Фрагмент якого способу запису алгоритму наведено нижче?

```

if (n > m)
    n = n - m;
else
    m = m - n;

```

- 1) Графічний спосіб.
 - 2) За допомогою псевдокодів.
 - 3) За допомогою мови програмування.
 - 4) Немає правильної відповіді.
7. Фрагмент якого способу запису алгоритму наведено нижче?



- 1) Графічний спосіб.
 2) За допомогою псевдокодів.
 3) За допомогою мови програмування.
 4) Немає правильної відповіді.
8. У якому із способів запису алгоритму виконавець сам визначає крок, з якого починати виконання алгоритму?
- 1) Графічний спосіб.
 2) За допомогою псевдокодів.
 3) За допомогою мови програмування.
 4) Немає правильної відповіді.
9. У якому із способів запису алгоритму виконавець випадковим чином визначає крок, на якому він може завершити виконання алгоритму?
- 1) Графічний спосіб.
 2) За допомогою псевдокодів.
 3) За допомогою мови програмування.
 4) Немає правильної відповіді.
10. Фрагмент якого способу запису алгоритму наведено нижче?

якщо $(a < x)$ **і** $(x \leq b)$

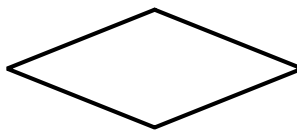
- 1) Графічний спосіб.
 2) За допомогою мови програмування Pascal.
 3) За допомогою мови програмування C#.
 4) Немає правильної відповіді.
11. При використанні якого способу може бути використано вказаний спосіб запису умови?

якщо $(a = b)$ **або** $(b = c)$

- 1) Графічний спосіб.
 2) За допомогою мови програмування Pascal.
 3) За допомогою мови програмування C#.
 4) Немає правильної відповіді.
12. Яким з наведених блоків позначають *початок та кінець алгоритму*?



1)



2)



3)

13. Яким з наведених блоків позначають *введення даних*?



1)



2)



3)

14. Яким з наведених блоків позначають *виведення даних*?



1)

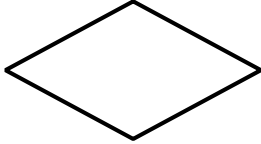


2)



3)

15. Який з наведених блоків використовують для перевірки умов?



1)



2)



3)

16. Який з наведених блоків використовується для звернення до допоміжних алгоритмів?



1)



2)



3)

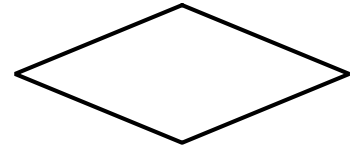
17. Який з наведених блоків використовується для запису циклу з параметрами?



1)

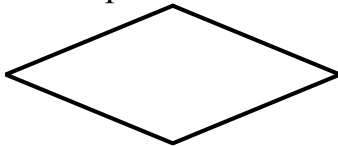


2)



3)

18. Який з наведених блоків використовується коли необхідно деякій величині присвоїти певне значення?



1)



2)



3)

19. Який блок при записі блок-схеми має дві лінії виходу?

- 1) початок;
- 2) умова;
- 3) процес;
- 4) кінець.

20. Скільки блоків



може бути використано при побудові алгоритму?

- 1) Тільки один.
- 2) Не менше трьох.
- 3) Не більше одного.
- 4) Немає правильної відповіді.

21. Скільки разів при розробці алгоритмів може зустрічатися блок

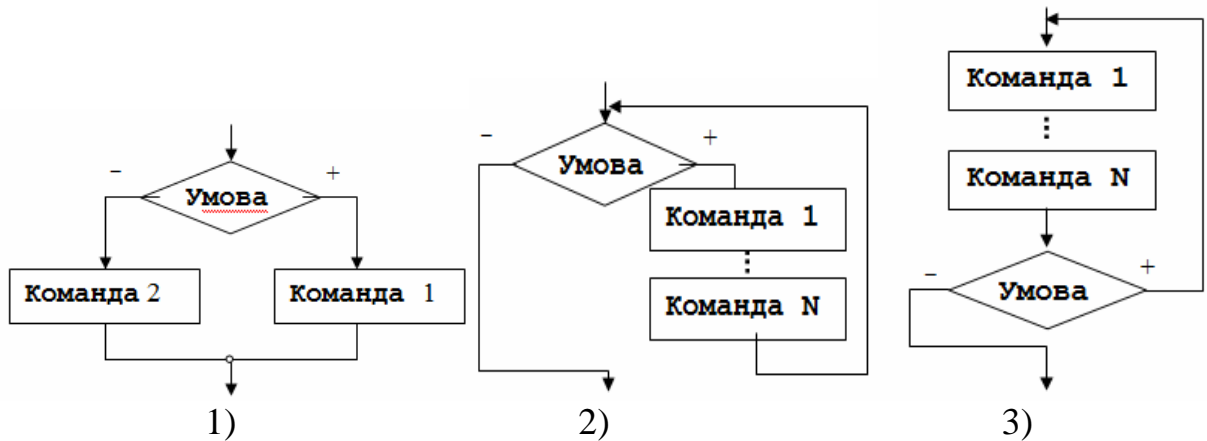


- 1) Тільки один.
- 2) Не менше трьох.

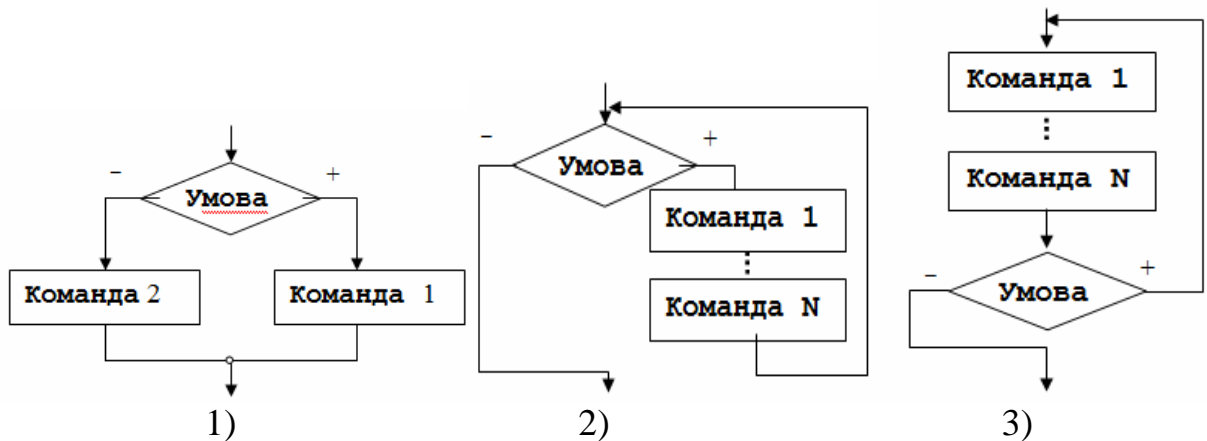
- 3) Не більше одного.
4) Немає правильної відповіді.

2.3. Базові структури управління

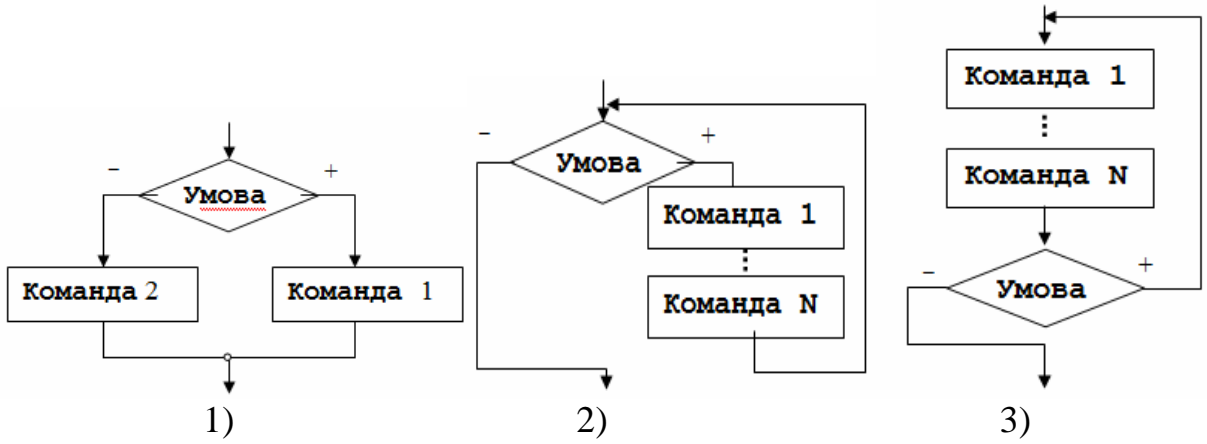
1. У якому з пунктів вказано *базові структури алгоритмів*?
- 1) Діалогові, допоміжні, лінійні.
 - 2) Слідування, розгалуження, повторення.
 - 3) Аргументи, результати, допоміжні величини.
2. За допомогою якої структури описують процес перетворення даних, що не вимагає перевірки жодних умов чи виконання повторюваних операцій?
- 1) Структури слідування.
 - 2) Структури повторення.
 - 3) Структури розгалуження.
2. Яка з наведених структур є структурою розгалуження?



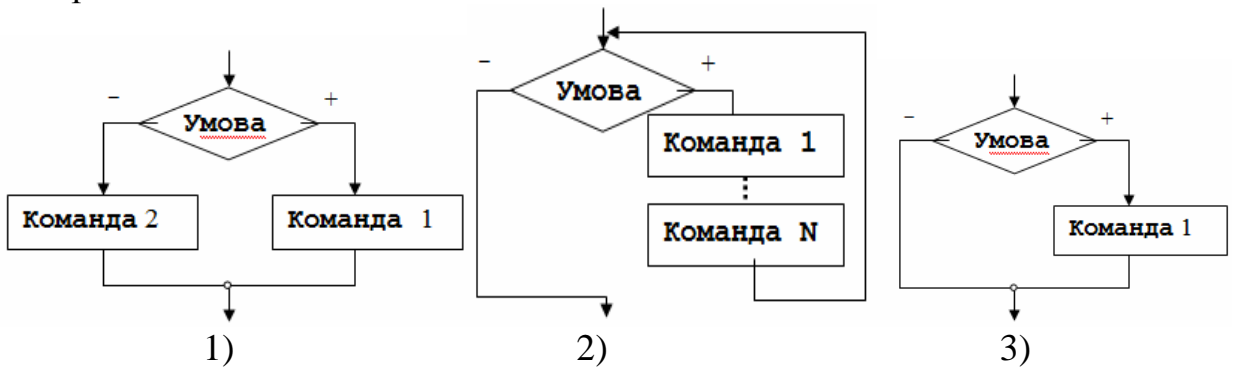
3. Яка з наведених структур є структурою повторення з передумовою?



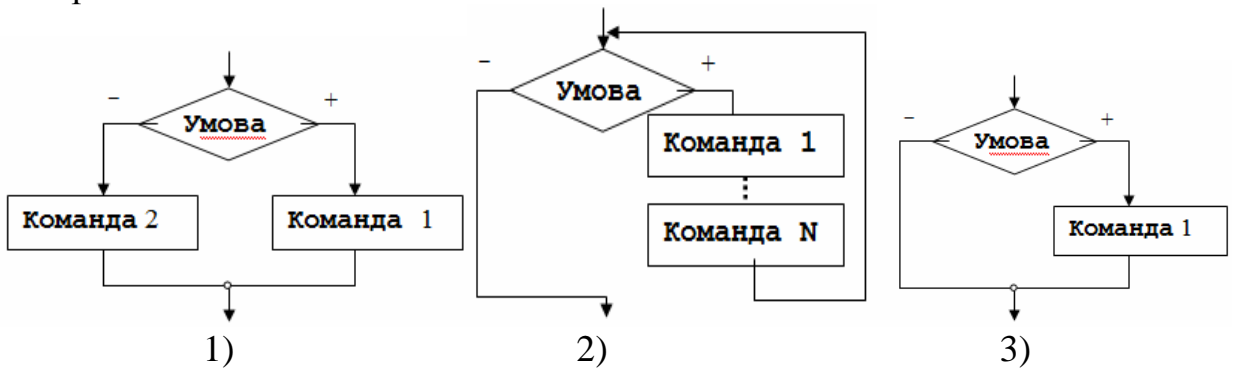
4. Яка з наведених структур є структурою повторення з післяумовою?



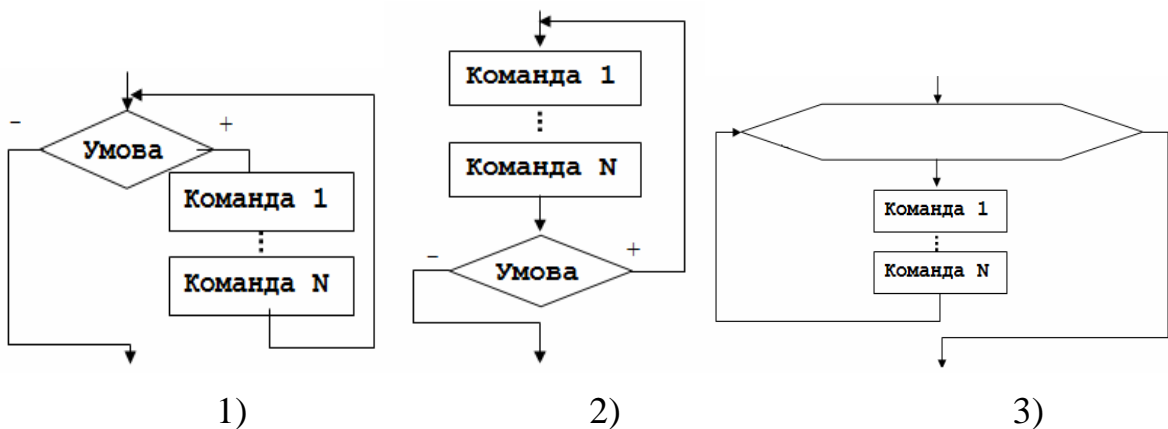
5. Яка із наведених структур є структурою розгалуження з однією альтернативою?



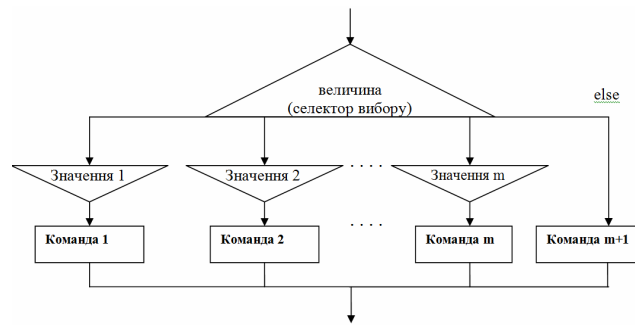
6. Яка із наведених структур є структурою розгалуження з двома альтернативами?



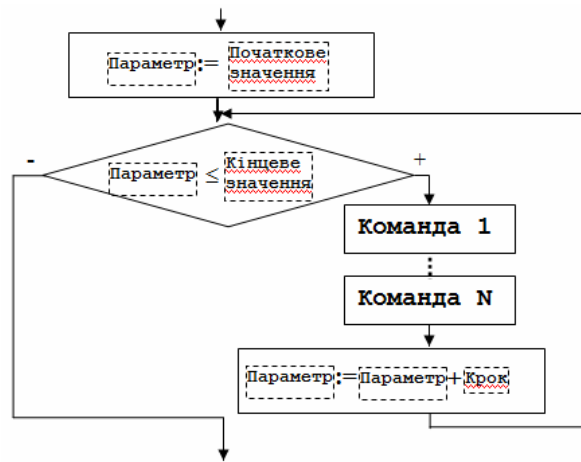
7. Яка із структур є структурою циклу з параметром?



6. Якою є подана структура?

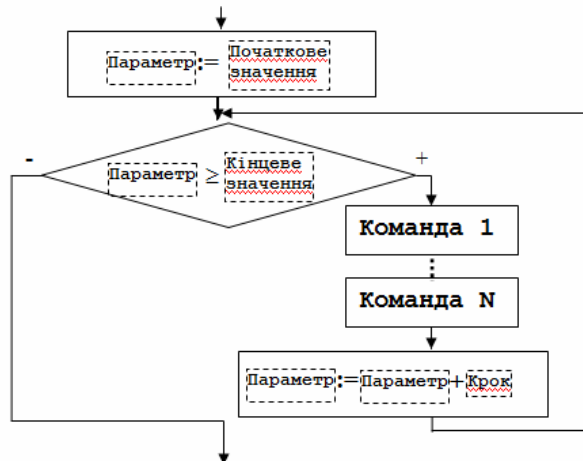


- 1) Структура слідування.
 - 2) Структура розгалуження.
 - 3) Структура циклу з параметром.
 - 4) Немає правильної відповіді.
7. При виконанні якого із циклів тіло циклу буде виконуватися принаймні один раз?
- 1) У циклі з передумовою.
 - 2) У циклі з післяумовою.
 - 3) У циклі з параметром.
 - 4) Немає правильної відповіді.
8. При побудові якого із циклів необхідно задати початкове, кінцеве значення та фіксований крок зміни величини?
- 1) У циклі з передумовою.
 - 2) У циклі з післяумовою.
 - 3) У циклі з параметром.
 - 4) Немає правильної відповіді.
9. При використанні якого типу циклу можлива ситуація, коли команди, що входять до тіла циклу, не будуть виконані жодного разу?
- 1) Цикл з передумовою.
 - 2) Цикл з післяумовою.
 - 4) Немає правильної відповіді.
10. Чи може цикл з параметром бути замінений за допомогою циклу з передумовою?
- 1) Так, завжди.
 - 2) Так, для деяких випадків.
 - 3) Не може.
 - 4) Немає правильної відповіді.
11. Яким є описаний нижче цикл?



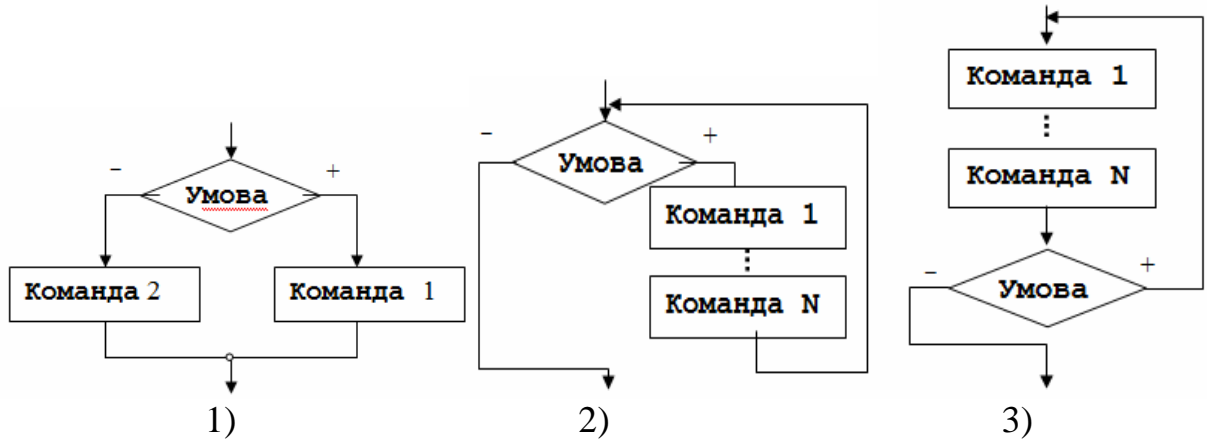
- 1) Цикл з параметром.
- 2) Цикл з передумовою.
- 3) Цикл з післяумовою.
- 4) Немає правильної відповіді.

12. Яким є крок у циклі з параметром, якщо його можна замінити наведеним нижчи циклом з передумовою?

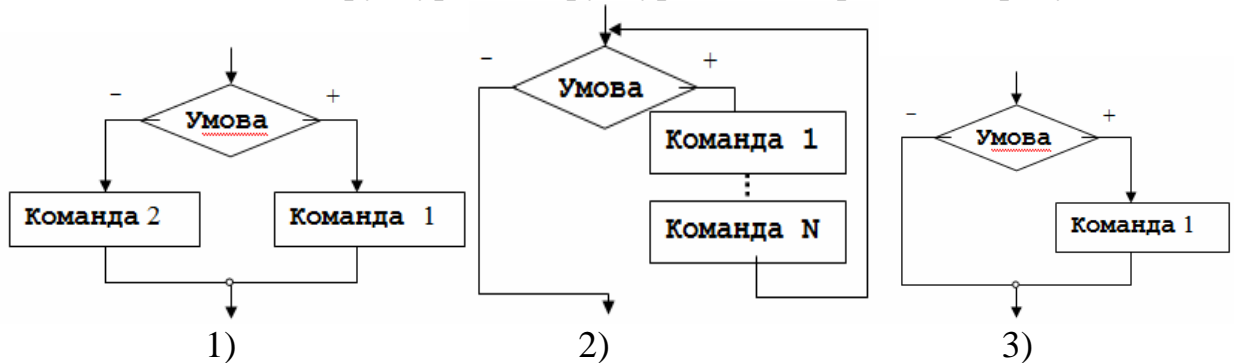


- 1) Додатним.
- 2) Від'ємним.
- 3) Рівним нулю.
- 4) Немає правильної відповіді.

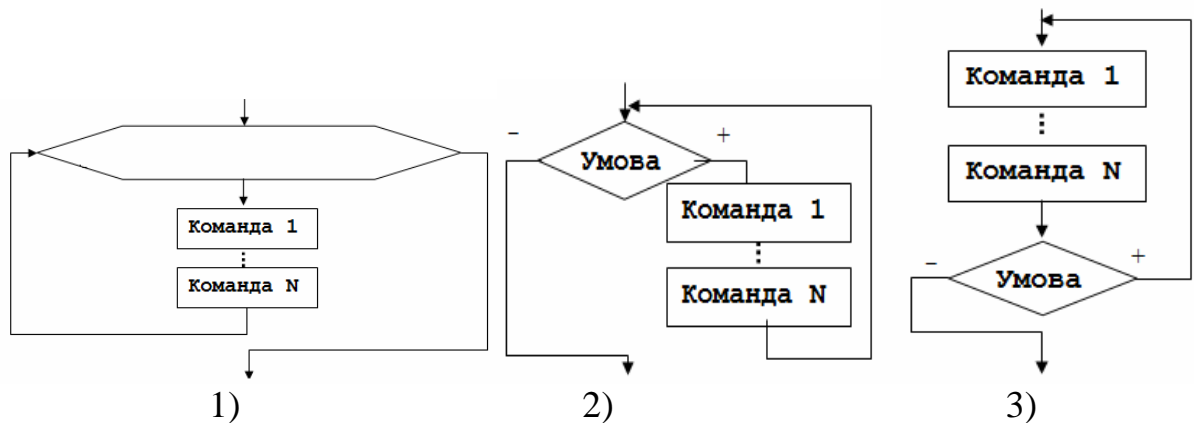
13. Яка з наведених структур не є структурою повторення?



14. Які з наведених структур не є структурами повторення з передумовою?



15. Які з наведених структур не є структурами повторення з післяумовою?



16. У якому із циклів умова продовження роботи циклу не може складатися із двох і більше умов, об'єднаних за допомогою службових слів «i» та «або»?

- 1) Цикл з передумовою.
- 2) Цикл з післяумовою.
- 3) Цикл з параметром.

17. Чи вірним є твердження?

Базові структури управління завжди мають один вхід і один вихід.

- 1) Так, завжди.
- 2) Ні. Вони можуть мати довільну кількість входів.
- 3) Ні. Вони можуть мати довільну кількість виходів.
- 4) Немає правильної відповіді.

18. Завершіть твердження. Робота циклу з передумовою завершується тоді, коли ...

- 1) перестане виконуватися умова продовження роботи циклу.

- 2) починає виконуватися умова продовження роботи циклу.
 3) Немає правильної відповіді.
19. Завершіть твердження. У циклі з параметром, якщо значення параметра дорівнює кінцевому значенню, то ...
- 1) цикл завершується.
 - 2) робота циклу буде продовжена.
 - 3) все залежить від знаку кроку.
 - 4) Немає правильної відповіді.
20. Скільки разів буде виконано тіло циклу з параметром (крок є більшим за нуль), якщо значення параметра більше кінцевого значенню.
- 1) один раз.
 - 2) не менше двох.
 - 4) Немає правильної відповіді.
21. Чи можна замінити структуру вибору за допомогою структур розгалуження?
- 1) Так, завжди.
 - 2) Так, для деяких задач.
 - 3) Ні. Ніколи.
 - 4) Немає правильної відповіді.

2.4. Типи алгоритмів

1. У залежності від структур, які використовуються при записі алгоритму, можна виділити такі **типи алгоритмів**:
- 1) лінійні, з розгалуженням та циклічні;
 - 2) лінійні, з розгалуженням;
 - 3) лінійні, циклічні;
2. Який алгоритм називається **лінійним**?
- 1) Це такий алгоритм, у якому виконуються всі без виключення команди послідовно відповідно до їх розміщення.
 - 2) Це такий алгоритм, у якому одні і ті самі вказівки (операції, оператори) виконуються багаторазово стосовно до різних значень змінних.
 - 3) Це такий алгоритм, у якому виконуються ті або інші вказівки залежно від результату перевірки деякої умови (або сукупності умов).
3. Який алгоритм називається алгоритмом з **роозгалуженням**?
- 1) Це такий алгоритм, в якому виконуються всі без виключення вказівки послідовно відповідно до їх розміщення.
 - 2) Це такий алгоритм, в якому одні і ті самі вказівки (операції, оператори) виконуються багаторазово стосовно до різних значень змінних.
 - 3) Це такий алгоритм, в якому виконуються ті або інші вказівки залежно від результату перевірки деякої умови (або сукупності умов).
4. Який алгоритм називається **циклічним**?
- 1) Це такий алгоритм, у якому виконуються всі без виключення вказівки послідовно відповідно до їх розміщення.
 - 2) Це такий алгоритм, у якому одні і ті самі вказівки (операції, оператори)

виконуються багаторазово стосовно до різних значень змінних.

3) Немає правильної відповіді.

5. Чи можна за допомогою базових структур управління зображати алгоритми будь-якої складності?

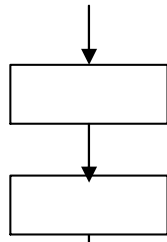
1) Так. Завжди

2) За виключенням класу особливих задач.

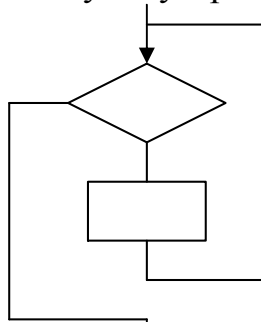
3) Ні.

4) Немає правильної відповіді.

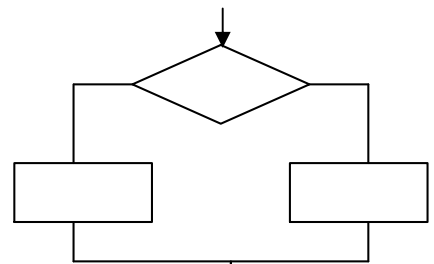
6. Які з наведених структур можуть зустрічатися у *лінійному алгоритмі*?



1)

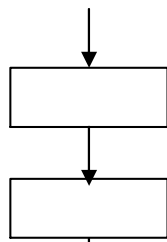


2)

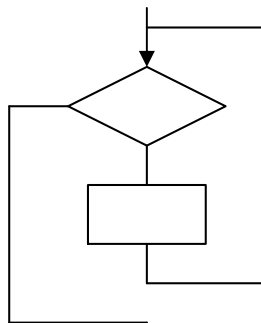


3)

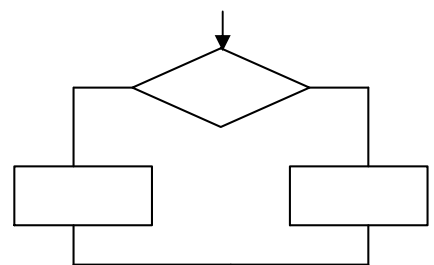
7. Які з наведених структур можуть зустрічатися у *алгоритмі з розгалуженням*?



1)

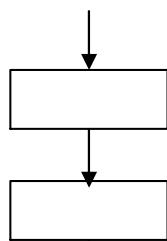


2)

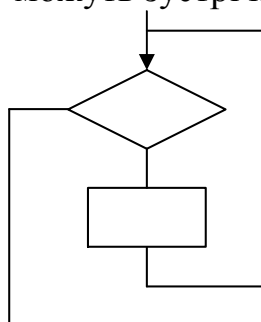


3)

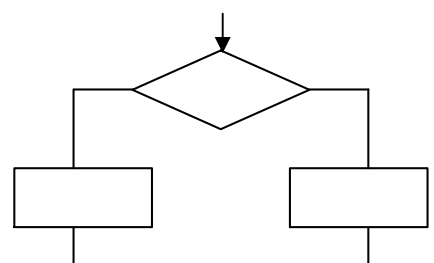
8. Які з наведених структур можуть зустрічатися у *циклічному алгоритмі*?



1)



2)



3)

7. У якому з алгоритмів використовуються лише структури слідування?

1) Лінійному.

2) З розгалуженням.

3) Циклічному.

8. Якого типу є наведений алгоритм вимірювання 1л води за допомогою банок ємністю 3л і 5 л.

Крок 1. Наповнити 3л банку.

Крок 2. Перелити воду з 3л банки у 5л банку.

Крок 3. Знову наповнити 3л банку.

Крок 4. Доповнити 5л банку водою з 3л. В результаті у 3л банці залишиться 1л води. Кінець алгоритму.

- 1) Циклічний.
- 2) Лінійний.
- 3) З розгалуженням.
- 4) Немає правильної відповіді.

9. Якого типу є наведений алгоритм, що дозволяє перевезти по одному через річку без втрат вовка, козу і капусту?

- Переправити на той берег козу, вовка залишити з капустою;
- Повернутись, взяти вовка, переправитись з ним до кози;
- Забрати козу і повернутись назад до капусти;
- Залишити козу, забрати і перевезти капусту до вовка;
- Повернутись і забрати козу.

- 1) Циклічний.
- 2) Лінійний.
- 3) З розгалуженням.
- 4) Немає правильної відповіді.

10. Якого типу є наведений алгоритм?

Крок 1. Дано два цілих додатних числа. Якщо вони рівні, то перше з них і є найбільшим спільним дільником, якщо ж ні, то перейдемо до кроку 2.

Крок 2. Порівняємо два числа і виберемо найбільше.

Крок 3. Більше з двох чисел замінимо різницею більшого і меншого.

Крок 4. Перейдемо до Кроку 1.

- 1) Циклічний.
- 2) Лінійний.
- 3) З розгалуженням.
- 4) Немає правильної відповіді.

11. Чи може бути алгоритм лінійним, якщо він містить вказаний фрагмент?

Крок 1. Дано два цілих додатних числа. Якщо вони рівні, то перше з них збільшуємо на 2.

Крок 2. Якщо друге число ділиться на 3 то множимо його на 6.

...

- 1) Так.
- 2) Залежить від значень чисел.
- 3) Немає правильної відповіді.

12. Якого типу є наведений алгоритм?

АЛГОРИТМ Найбільший спільний дільник**ПОЧАТОК****ВВЕДЕННЯ** «Задайте два додатних цілих числа» n, m **ПОКИ** $n \neq m$ **ПОЧАТОК****ЯКЩО** $n > m$ **ТО** $n := n - m$ **ІНАКШЕ** $m := m - n$ **ВСЕ****КІНЕЦЬ****ВИВЕДЕННЯ** «Найбільший спільний дільник даних чисел:» n **КІНЕЦЬ**

- 1) Циклічний.
 - 2) Лінійний.
 - 3) З розгалуженням.
 - 4) Немає правильної відповіді.
13. Завершіть твердження. Алгоритм знаходження периметру трикутника, заданого координатами вершин на площині є...
- 1) лінійним;
 - 2) з розгалуженням;
 - 3) циклічним;
 - 4) немає правильної відповіді.
14. Завершіть твердження. Алгоритм розв'язання довільного квадратного рівняння є...
- 1) лінійним;
 - 2) з розгалуженням;
 - 3) циклічним;
 - 4) немає правильної відповіді.
14. Завершіть твердження. Алгоритм встановлення типу трикутника, заданого довжинами сторін є...
- 1) лінійним;
 - 2) з розгалуженням;
 - 3) циклічним;
 - 4) немає правильної відповіді.
15. Завершіть твердження. Алгоритм знаходження радіуса описаного кола навколо трикутника, заданого довжинами сторін є...
- 1) лінійним;
 - 2) з розгалуженням;
 - 3) циклічним;
 - 4) немає правильної відповіді.
16. Завершіть твердження. Алгоритм знаходження найбільшого серед трьох дійсних чисел є...
- 1) лінійним;

- 2) з розгалуженням;
3) немає правильної відповіді.
17. Завершіть твердження. Алгоритм знаходження середнього арифметичного n дійсних чисел є...
- 1) лінійним;
2) з розгалуженням;
3) циклічним;
4) немає правильної відповіді.
18. Завершіть твердження. Алгоритм знаходження найбільшого серед n дійсних чисел є...
- 1) лінійним;
2) з розгалуженням;
3) циклічним;
4) немає правильної відповіді.
19. За допомогою алгоритмів яких типів можна розв'язати задачу знаходження факторіалу $n!$.
- 1) лінійного
2) циклічного;
3) немає правильної відповіді.
20. Завершіть твердження. Алгоритм знаходження найменшого серед n дійсних чисел є...
- 1) лінійним;
2) з розгалуженням;
3) циклічним;
4) немає правильної відповіді.
21. Завершіть твердження. Алгоритм порівняння двох прямокутників на рівність (прямокутники задано довжинами сторін) є ...
- 1) лінійним;
2) розгалуженим;
3) циклічним;
4) немає правильної відповіді.

2.5. Структурний підхід до розробки алгоритмів.

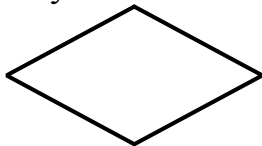
Допоміжні алгоритми

1. Суть структурного підходу за методикою «зверху-вниз» визначається такими засадами:

- 1) складна задача розбивається на простіші, що легко сприймаються, частини, кожна з яких має один вхід і один вихід;
2) складна задача розбивається на простіші, що легко сприймаються, частини, кожна з яких має декілька входів і декілька виходів;
3) складна задача розбивається на простіші, що легко сприймаються, частини, кожна з яких має декілька входів і один вихід;
4) Немає правильної відповіді.

2. Згідно із теоремою про структурування, будь-який правильний алгоритм можна подати за допомогою наступних базисних структур алгоритмів:
- 1) слідування і розгалуження;
 - 2) розгалуження, слідування і повторення(циклу);
 - 3) розгалуження і повторення(циклу);
 - 4) повторення(циклу) і розгалуження.
3. Завершіть твердження. Алгоритм повинен мати ...
- 1) один вхід та один вихід, не містити нескінченних циклів і невиконуваних вказівок (команд);
 - 2) один вхід та декілька виходів, не містити нескінченних циклів і невиконуваних вказівок (команд);
 - 3) один вхід та один вихід, містити нескінченні цикли і невиконувани вказівки (команди);
 - 4) декілька входів та один вихід, не містити нескінченних циклів і невиконуваних вказівок (команд).
4. Завершіть означення. *Допоміжним алгоритмом* називають:
- 1) алгоритм, який створений наперед і викликається та повністю виконується всередині іншого, основного алгоритму;
 - 2) алгоритм, який створений наперед, починається та завершується разом з основним алгоритмом;
 - 3) алгоритм, який створюється в процесі роботи основного алгоритму;
 - 4) Немає правильної відповіді.
5. Завершіть твердження. *Формальні параметри* – це ...
- 1) величини, що описано у заголовку допоміжного алгоритму;
 - 2) величини, які необхідно задати при виконанні основного алгоритму;
 - 3) проміжкові величини, що використовуються у допоміжному алгоритмі;
 - 4) немає правильної відповіді.
6. Завершіть твердження. *Фактичні параметри* – це ...
- 1) параметри, описані у заголовку допоміжного алгоритму.
 - 2) параметри, що є результатами роботи допоміжного алгоритму.
 - 3) параметри, що є вказуються при виклику допоміжного алгоритму.
 - 4) Немає правильної відповіді.
7. Які величини можуть бути вказані при описі формальних параметрів алгоритму?
- 1) Величини, необхідні для роботи допоміжного алгоритму.
 - 2) Величини, що є результатами роботи допоміжного алгоритму
 - 3) Проміжкові величини.
8. Завершіть твердження. В якості фактичних параметрів використовують...
- 1) значення величин, необхідних для роботи допоміжного алгоритму.
 - 2) величини, у яких необхідно зберегти результати роботи допоміжного алгоритму.
 - 3) величини, що не можуть бути ні аргументами, ні результатами.
9. Якими із способів можна здійснити повернення результатів із допоміжного алгоритму?

- 1) Через ім'я допоміжного алгоритму.
 2) Через формальні параметри.
 3) Немає правильної відповіді.
10. Які з тверджень мають місце по відношенню до структурного підходу до розробки алгоритму?
 1) Не можна використовувати структури слідування, розгалуження і циклу.
 2) Передбачає розбиття складної задачі на більш прості підзадачі.
 3) Немає правильної відповіді.
11. Завершіть означення. *Рекурсивний алгоритм* – це ...
 1) алгоритм, що містить вкладені цикли.
 2) алгоритм, що містить вкладені структури розгалуження.
 3) алгоритм, що не містить вкладених циклів.
 4) Немає правильної відповіді
12. Яку кількість результатів можна повернути через ім'я допоміжного алгоритму?
 1) Довільну кількість.
 2) Один.
 3) Жодного.
 4) Залежить від типу основного алгоритму.
13. Чи можна одночасно повертати результати роботи допоміжного алгоритму через ім'я допоміжного алгоритму і через формальні параметри?
 1) Так.
 2) Ні. Не можна.
 3) Залежить від типу основного алгоритму.
 4) Немає правильної відповіді.
14. Який із блоків використовується при описі заголовку допоміжного алгоритму?



1)



2)

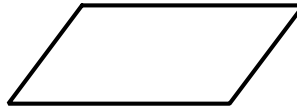


3)

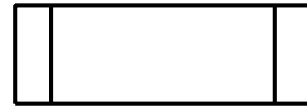
15. Який із блоків використовується при звертанні до допоміжного алгоритму?



1)



2)



3)

16. Яку кількість результатів можна повернути із допоміжного алгоритму, використовуючи формальні параметри?

- 1) Довільну кількість.
 2) Один.
 3) Жодного.
 4) Залежить від типу основного алгоритму.

17. Яким чином описують величину, яка є проміжковою у допоміжному алгоритмі?
- 1) Як формальний параметр.
 - 2) Як фактичний параметр.
 - 3) Немає правильної відповіді.
18. Завершіть твердження. Особливістю рекурсивного алгоритму є те, що ...
- 1) він не може містити виклик інших допоміжних алгоритмів.
 - 2) він може містити виклик як допоміжного алгоритму самого себе з іншими вхідними даними.
 - 3) Немає правильної відповіді.
19. Використовуючи якого типу алгоритми може бути обчислено факторіал числа n ?
- 1) Рекурсивний.
 - 2) З розгалуженням.
 - 3) Циклічний.
 - 4) Немає правильної відповіді.
20. Використовуючи якого типу алгоритми може бути обчислено n -тє число Фібоначчі?
- 1) Рекурсивний.
 - 2) З розгалуженням.
 - 3) Циклічний.
 - 4) Немає правильної відповіді.
21. Як можна повернути корені квадратного рівняння, як результати роботи допоміжного алгоритму.
- 1) Через ім'я допоміжного алгоритму.
 - 2) Через формальні параметри.
 - 3) Залежить від знаку коренів.

Список використаної літератури

1. Білоусова Л.І., Вепрік С.А., Муравка А.С. Збірник задач по курсу інформатики.-Х.: Світ дитинства, 2000.
2. Варлань А.Ф., Апатова Н.В. Інформатика: Підр. Для учнів 10-11 кл. серед.загальноосвіт.шк. – К.: Форум, 2001.
3. Забарна А.П. Основи алгоритмізації та програмування. Інтерактивні технології навчання на уроках. – Тернопіль. Мандрівець, 2006.
4. Караванова Т.П. Інформатика: основи алгоритмізації та програмування: 777 задач з рекомендаціями та прикладами: Навч. посіб. для 8-9 кл. із поглибл. вивч. інф-ки – К.: Генеза. – 2006.- 286 с.
5. Книга вчителя інформатики: Довідково-методичне видання / Упоряд. Н.С.Прокопенко, Т.Г.Проценко – Харків: ТОРСІНГ ПЛЮС, 2005.– 256с.
6. Кнут Д. Искусство программирования. — М.: Вильямс, 2000
7. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МЦНМО, 2001. — 960 с., 263 ил.
8. Р.І.Пиртко, І.А.Сеньків Уроки з основ інформатики та обчислювальної техніки. Основи алгоритмізації та програмування. Навчальний посібник. – Тернопіль. Навчальна книга – Богдан, 2007.
9. Реєстр програмних засобів навчального призначення // Інформатика та інформаційні технології в навчальних закладах. – 2006, №1. – С.180-189.

ЗМІСТ

1. ТЕОРЕТИЧНІ ОСНОВИ	3
1.1. Поняття алгоритму	3
1.2. Способи запису алгоритмів	7
1.3. Базові структури управління	11
1.4. Типи алгоритмів.....	15
1.4.1. Лінійні алгоритми.....	15
1.4.2. Алгоритми з розгалуженням	17
1.4.3. Циклічні алгоритми.....	22
1.5. Структурний підхід до розробки алгоритмів. Допоміжні алгоритми...	25
2. РОЗРОБКИ ТЕСТІВ.....	38
2.1. Основні поняття та означення.....	38
2.2. Способи запису алгоритмів	41
2.3. Базові структури управління	44
2.4. Типи алгоритмів.....	49
2.5. Структурний підхід до розробки алгоритмів. Допоміжні алгоритми...	53
Список використаної літератури	57

Відповідальний за випуск: завідувач кафедрою системного аналізу і теорії
оптимізації к. ф.-м. н., доц. Кузка О.І.

Автори: к. т. н., доц. Семйон І.В.,
к. ф.-м. н., доц. Чупов С.В.,
к. ф.-м. н., доц. Брила А.Ю.,
к. ф.-м. н., Антосяк П.П.,
Дудла М.В.

Рецензенти: к.ф.-м.н., доц. Погоріляк О.О.

ОСНОВИ АЛГОРИТМІЗАЦІЇ

Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного
факультету спеціальності "прикладна математика"