Практичне заняття № 1

Наукове пізнання навколишнього світу людина здійснює за допомогою його моделювання. Кожна наукова дисципліна широко використовує різні моделі. Графічні схеми, математичні рівняння, діаграми причинно-наслідкових зв'язків, таблиці і навіть вербальні конструкції є різновидами моделювання.

Модель - це формалізоване, спрощене представлення реального об'єкту

Будь-яка наукова модель має цільовий характер, тобто вона будується для конкретних цілей і для вирішення певних задач. Вона достатньо точно відображає ті сторони модельованого явища, які мають ключове значення для вирішення поставленої задачі. Укладена в моделі помилка рано чи пізно заважає вирішувати інші задачі, пов'язані з модельованим об'єктом. Вирішити нові задачі можна або шляхом уточнення, узагальнення первинної моделі, або шляхом побудови нової моделі.

Алгоритм – фундаментальне поняття математики, йому не можна дати точне математичне визначення із залученням інших фундаментальних понять. Відсутність такого визначення не заважає інтуїтивному розумінню його змісту. Але в літературних джерелах наводяться різні варіанти розуміння сутності алгоритму. В загальному випадку під терміном "алгоритм" розуміють шлях (метод, спосіб) розв'язання задачі. Таке тлумачення не дозволяє досліджувати особливості роботи, конструювати ефективні алгоритми. Проблема розв'язується застосуванням моделей алгоритму.

В залежності від складності задачі, мети дослідження, практичного застосування моделі поділяються на декілька груп, які показані на *рис.1*.

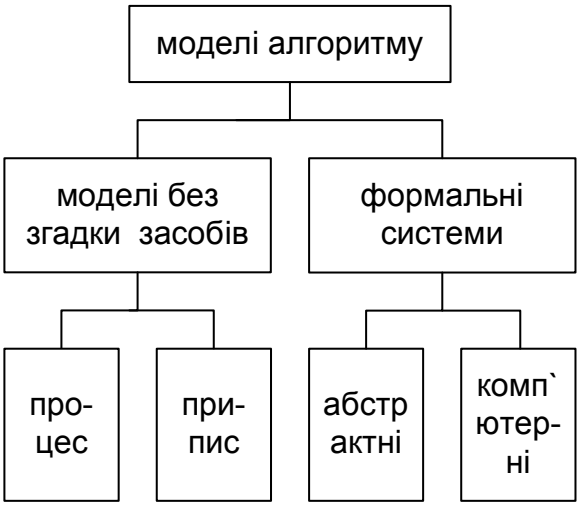


Рис. 1

Виконанню обчислювальних операцій без вказівки на засоби виконання відповідають дві моделі: "алгоритм – процес" і "алгоритм – припис"

Першу модель "**алгоритм – процес**" виконання чотирьох арифметичних операцій детально у словесній формі описав аль-Хорезмі (IX стор). Пізніше в працях Хр. Рудольфа (XVI стор.) і Г.В.Лейбніця (XVII стор.) модель аль-Хорезмі набула наступного тлумачення – *"Алгоритм означає будь-який регулярний обчислювальний процес, який за кінцеву кількість кроків розв'язує задачі визначеного класу*". Це тлумачення близьке до наведеного у сучасній фундаментальній монографії: "*Кажучи неформально, алгоритм – це будь-яка коректно сформульована обчислювальна процедура, на вхід якої подається деяка величина або набір величин, і результатом виконання якої є вихідна величина або набір значень*". Більш детальне тлумачення з описом властивостей алгоритму дано А Марковим:

“а) Алгоритм – це процес послідовної побудови величин, який проходить в дискретному часі таким чином, що в початковий момент задається початкова скінчена система величин, а в кожний наступний момент система величин отримується за певним законом (програмою) із системи величин, які були в попередній момент часу (дискретність алгоритму).
б) Система величин, які утримуються в якийсь (не початковий) момент часу, однозначно визначається системою величин, отриманих в попередні моменти часу (детермінованість алгоритму).
в) Закон отримання наступної системи величин із попередньої повинен бути простим і локальним (елементарність кроків алгоритму).
г) Якщо спосіб отримання наступної величини із якої-небудь заданої величини не дає результату, то повинно бути вказано, що потрібно рахувати результатом алгоритму (спрямованість алгоритму).
д) Початкова система величин може вибиратися із деякої потенційно нескінченної множини (масовість алгоритму)”.

Це розширене тлумачення можна знайти в ряді сучасних джерел. Порівнюючи ці три тлумачення, не можна сказати, яке з них краще або вірніше, бо це лише неформальний опис моделі алгоритмічного процесу. Для такого опису не має математичних підстав порівняння. Розглянуті тлумачення відрізняються один від одного. Наприклад, процес вимірюється часом виконання, припис вимірюється кількістю інструкцій програми або блок-схем програми, але ці відміни незначні.

Друга модель “**алгоритм – припис**” реалізується у вигляді блок-схем програм і програм на мові високого рівня. Прикладом тлумачення алгоритму на основі цієї моделі є: “*Алгоритм – це послідовність інструкцій для виконання деякого завдання*”.

Наступні варіанти тлумачення зумовлені використанням конструктивно заданих математичних моделей алгоритму – формальних алгоритмічних систем. Абстрактні формальні системи використовуються для дослідження теоретичних проблем обчислень, наприклад, проблем розв’язності. На основі абстрактних систем була створена теорія складності. До формальних алгоритмічних систем відносяться два класи математично строго визначених моделей, які містять всі основні засоби для здійснення обчислювального процесу. До першого класу відносяться абстрактні моделі, які у своєму визначенні не описують апаратні засоби, хоча їх наявність припускається. Також припускається, що операції алгоритмічного процесу здійснюються людиною без будь-яких інтелектуальних зусиль. Прикладами моделей цього класу є машина Тюрінга, нормальні алгоритми Маркова та декілька інших. Тлумаченням алгоритму, що випливає з аналізу цих моделей є: “*Алгоритм – точний припис, який задає обчислювальний процес (що називається в цьому випадку алгоритмічним), що починається з довільного початкового даного (з деякої сукупності можливих для даного алгоритму початкових даних) і спрямований на отримання результату, який повністю визначається цим початковим даним*”. Тут точний припис і алгоритмічний процес об’єднані в одному тлумаченні. Абстрактні формальні системи використовуються для дослідження теоретичних проблем обчислень, наприклад, проблем розв’язності. На основі абстрактних систем була створена теорія складності.

Другий клас формальних систем складається з моделей комп’ютерних алгоритмів. У цих моделях апаратні засоби задекларовані безпосередньо у її визначенні. Прикладом моделі другого класу є SH-модель алгоритму (SH – Software/Hardware). Теорія складності комп’ютерних алгоритмів суттєво відрізняється від абстрактної теорії. Тут крім технічних характеристик складності (часової, апаратної та ємнісної) використовуються додатково інформаційні (програмна та структурна). Модель комп’ютерного алгоритму дозволила формально визначити властивість “елементарність”, сформулювати властивість “ієрархічність”, створити модель універсального обчислювача. Відображення змісту моделі комп’ютерного алгоритму знайшло в неформальному тлумаченні.: *“Алгоритм - це*

фіксована для розв'язання деякого класу задач конфігурація апаратно-програмних засобів перетворення, передавання і зберігання даних, який задає обчислювальний процес (що називається в цьому випадку алгоритмічним), який починається з будь-яких початкових даних (з деякої потенційно нескінченної сукупності можливих для даного алгоритму початкових даних) і скерований на отримання результату, повністю визначеного цими початковими даними".

Таким чином кожне наведене тлумачення поняття "алгоритм" адекватне обраній для дослідження конкретної моделі обчислень.

Практичне заняття № 2

Те, що зараз ми розуміємо під словом алгоритм, використовувалося в глибокій давнині, наприклад, теорема про залишки в Китаї (Китайська теорема), арифметичні операції в Індії. Але праці Евкліда і аль-Хорезмі для теорії складності алгоритмів мають особливе значення.

Мухамед ібн Муса з Хорезму, за арабським ім'ям – аль-Хорезмі (походженням з середньоазіатського міста Хорезм), видатний багдадський вчений, що працював у IX столітті н.е. У своїй книжці – трактаті "Про індійський рахунок" аль-Хорезмі описав десяткову систему числення і арифметичні операції " множення і ділення, сумування, віднімання та інші". Сьогодні збереглися лише переклади трактату. Перші з них відносяться до початку XII століття.

Далі ми наводимо цитати з "Книги про індійський рахунок", переклад якого був виконаний Ю.Копелевич з середньовічного латинського тексту, що зберігається у Кембриджському університеті.

Кожний розділ трактату, а іноді навіть абзац, починався словами "Сказав Альгорізмі…". Це словосполучення використовували у своїх лекціях і професори середньовічних університетів. Поступово ім'я аль-Хорезмі набуло звучання "алгоризм", "алгоритм" і навіть перетворилися у назву нової арифметики. Пізніше термін "алгоритм" почав означати регулярний арифметичний процес (Хр. Рудольф, 1525р.). І тільки наприкінці XVII ст. в роботах Лейбніца цей термін набув змістовності, яка не заперечує сучасному тлумаченню: "Алгоритм - це будь-який регулярний обчислювальний процес, що дозволяє за кінцеву кількість кроків розв'язувати задачі визначеного класу". Зауважимо, що за довгу еволюцію слова "алгоритм" було втрачено джерело його виникнення. І тільки у 1849 році сходознавець Ж. Рейно повернув нам ім'я аль-Хорезмі .

Зауважимо, що й слово "алгебра" бере свій початок з математичного трактату аль-Хорезмі "Книга відновлення і протиставлення". Мухамеду ібн Мусі ще не були відомі від'ємні числа, тому в процесі обчислень він користувався операцією перенесення від'ємника з одної частини рівняння в іншу, де той стає доданком. Цю операцію Мухамеда ібн Муса називав "відновленням". Слову "протиставлення" відповідає зміст збирання невідомих на одну сторону рівняння. Арабською "відновлення" – аль-джебр. Звідси походить слово "алгебра". Слова "алгоритм" і "алгебра" на перших кроках розвитку математики було щільно пов'язані між собою і за змістом і за походженням. Вони виникли з одного джерела і разом пройшли багатовіковий шлях еволюції, зайняли провідні місця у сучасній науковій термінології.

Здавна найбільшу увагу приділяли дослідженням алгоритму з метою мінімізації обсягу досліджень – часовій складності розв'язання задач. Але зміст складності алгоритму не обмежується однією характеристикою. В ряді випадків не менше значення має складність логіки побудови алгоритму, різноманітність його операцій, зв'язаність їх між собою. Ця характеристика алгоритму називається програмною складністю. В теорії алгоритмів, крім часової та програмної складності, досліджуються також інші характеристики складності, наприклад, ємкістна, але найчастіше розглядають дві з них - часову і програмну. Якщо у кінцевому результаті часова складність визначає час розв'язання задачі, то програмна складність характеризує ступінь інтелектуальних зусиль, що потрібні для синтезу алгоритму. Вона впливає на витрати часу проектування алгоритму.

Вперше значення зменшення програмної складності продемонстрував аль-Хорезмі у своєму трактаті "Про індійський рахунок". У часи аль-Хорезмі для розрахунків користувалась непозиційною римською системою числення. Її вузловими числами є I, V, X, L, C, D, M, всі решта чисел утворюються сумуванням і відніманням вузлових. аль-Хорезмі, мабуть, першим звернув увагу на складність римської системи числення у порівнянні з позиційною десятковою з точки зору простоти операцій, їх послідовного виконання та засвоєння. Він писав: "…ми вирішили розтлумачити про індійський рахунок за допомогою .IX. літер, якими вони виражали будь-яке своє число для *легкості і стислості*, *полегшуючи* справу тому, хто вивчає арифметику, тобто число найбільше і найменше, і все, що є в ньому від множення і ділення, сумування, віднімання та інше.". Виокремленні слова – *легкість, стислість*, *полегшення* свідчать перш за все про те, що

мова йде про програмну складність алгоритмів арифметичних операцій з використанням двох систем числення. Мабуть, ці слова аль-Хорезмі про складність алгоритмів при їх порівнянні були першими в історії арифметики.

Алгоритми реалізації арифметичних операцій, описані аль-Хорезмі у словесній формі, були першими у позиційній десятковій системі числення. Цікаво спостерігати, як точно і послідовно описує він алгоритм сумування, користуючись арабською системою числення і кільцем (нулем). Наведемо повністю цей алгоритм.

"Сказав Алгорізмі: Якщо ти хочеш додати число до числа або відняти число від числа, постав обидва числа в два ряди, тобто одне над другим, і нехай буде розряд одиниць під розрядом одиниць і розряд десятків під розрядом десятків. Якщо захочеш скласти обидва числа, тобто додати одне до другого, то додай кожний розряд до розряду того ж роду, який над ним, тобто одиниці до одиниць, десятки до десятків. Якщо в якому-небудь із розрядів, тобто в розряді одиниць або десятків, або якому-небудь іншому набереться десять, став замість них одиницю і висувай її в верхній ряд, тобто, якщо ти маєш в першому розряді, який є розряд одиниць, десять, зроби з них одиницю і підніми її в розряд десятків, і там вона буде означати десять. Якщо від числа залишилось що-небудь, що нижче десяти, аби якщо саме число нижче десяти, залиши його в тому ж розряді. А якщо нічого не залишиться, постав кружок, щоби розряд не був пустим; але нехай буде в ньому кружок, який займе його, аби не сталося так, що якщо він буде пустим, розряди зменшаться і другий буде прийнятий за перший, і ти обманешся в своєму числі. Те ж саме ти зробиш у всіх розрядах. Подібним же чином, якщо збереться у другому розряді .X., зробиш з них одиницю і піднімеш її в третій розряд, і там вона буде означати сто, а що лишається нижче .X., залишиться тут. Якщо ж нічого в інших не залишається , ставиш тут кружок, як вище. Так ти зробиш в інших розрядах, якщо буде більше".

Можна бачити, що в цьому опису є всі параметри алгоритму. Це один з перших відомих у світі вербальних арифметичних алгоритмів.

Розглянемо логіку побудови арифметичних процедур з використанням римської та арабської систем числення з метою порівняння їх за програмною складністю. Розглянемо приклад операції сумування. Будемо користуватися алгоритмом на основі табличного методу. У арабській системі з порозрядними операціями розмір таблиці 10*10. Визначення суми чергових розрядів двох чисел, наприклад, 2 і 3 за таблицею дорівнює 5, або 7 і 9 дорівнює 16. Ці таблиці ми пам'ятаємо з дитинства.

Інша ситуація є з римською системою числення. Крім таблиці (I,II,…X)*(I,II,…X), що є еквівалентом таблиці 10*10 у арабській позиційній системі, додатково потрібно ще чотири таблиці з вузловими числами римської системи L,C,D,M та доповнення табличного методу логічними процедурами. Наприклад, для операції сумування двох чисел CMLIX + XCIV потрібні таблиці більшого об'єму, ніж 10*10 кожна. Один з варіантів рахунку полягає в представленні чисел, по-перше, розділеними на окремі цифри, по-друге, проведенням операцій віднімання і сумування окремих цифр з використанням таблиць, по-третє, об'єднанням цифр, що залишилися, в єдине число.

CMLIX + XCIV = (-C) + M +L +(-I) + X + (-X) + C +(-I) + V;

Оскільки –C + C = 0; –X + X = 0; – I – I = – II, V – II = III,

то: CMLIX + CIV = M + L + III = MLIII;

Як бачимо, для проведення розрахунків у римській системі необхідно виконувати більше типів операцій, ніж у десятковій арабській позиційній системі числення. Крім того, у римській системі потрібні додаткові логічні перетворення, що суттєво ускладнюють зв'язки між окремими операціями обчислювального процесу.

Часова складність операцій сумування і віднімання в десятковій системі визначається кількістю розрядів у взаємодіючих числах. У римській системі часова складність залежить від порядку розташування цифр у числах. У тих випадках, коли не потрібно утворювати ланцюги логічних операцій, часова складність не перевищує часову складність операцій з десятковими числами. У протилежному випадку, як у розглянутому прикладі, зростання невелике.

Таким чином, за логікою побудови і різноманітністю операцій – програмною складністю, арабська система суттєво простіша за римську, що й довів аль-Хорезмі. За часовою складністю вони майже однакові.

Величезне революційне значення мало використання десяткової систем числення у Європі у всіх сферах життя – побуті, навчанні, торгівлі, суднобудуванні, техніці, географії, астрономії та багато інших. І те, що цей процес співпав з епохою Відродження, не є випадковим.

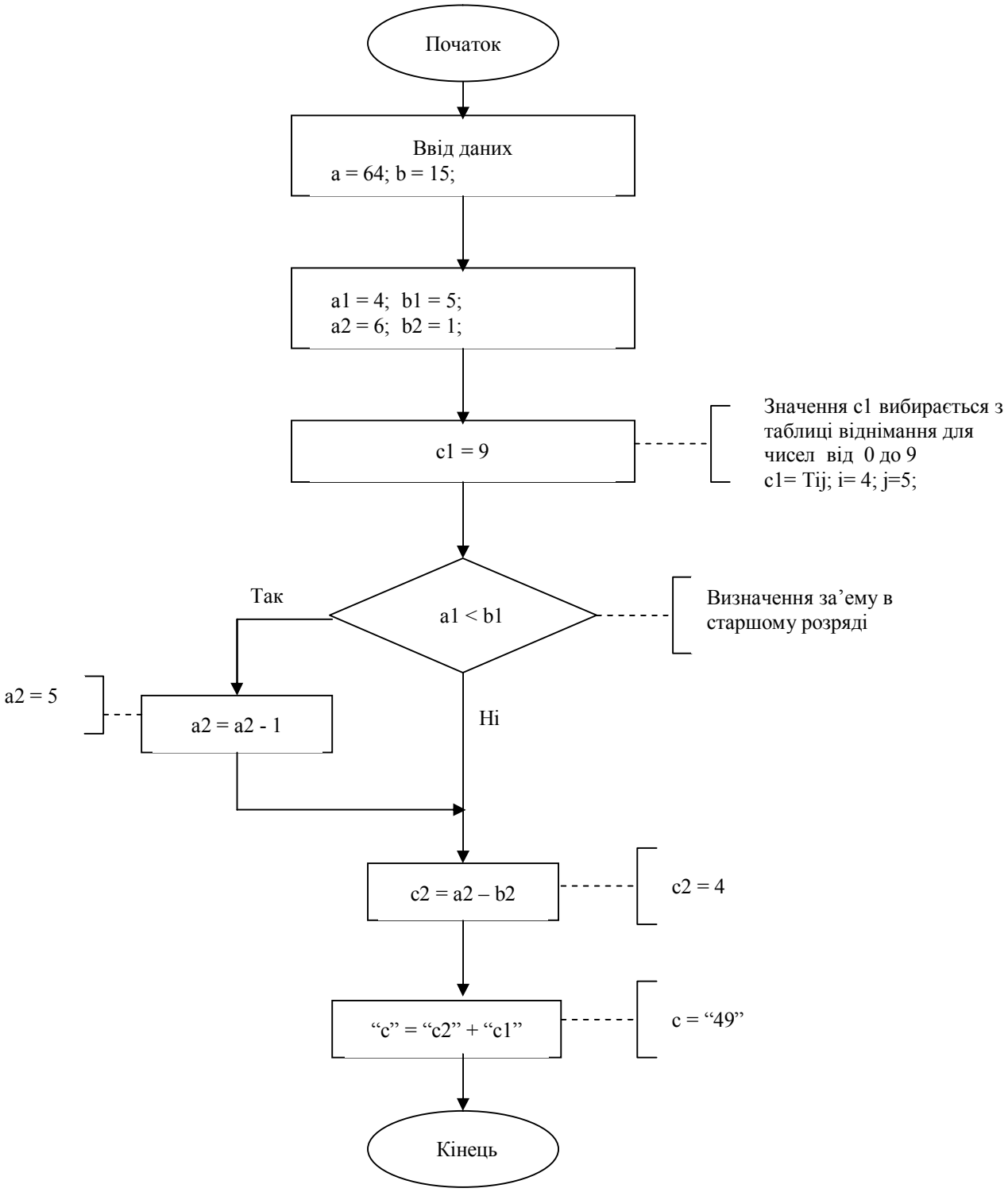
Введення десяткової системи числення, можливо, відіграло вирішальну роль прискорювача у поступових процесах Ренесансу. Значна роль в цьому належить видатному арабському вченому аль-Хорезмі.

1. *Мухаммад ибн Муса ал-Хорезми. Математические трактаты.- Ташкент, 1983.*
2. *2.Юшкевич А.П. История математики в средние века. – М1961.*

Приклад.

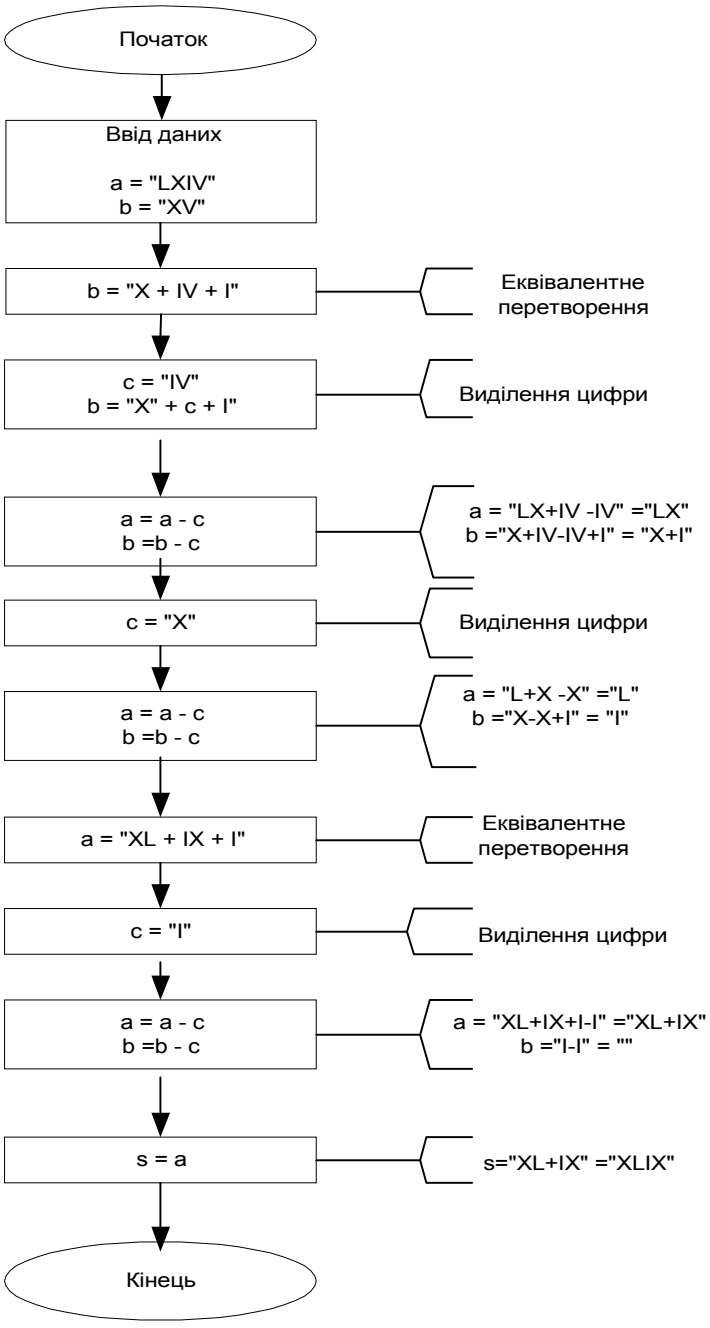
LXIV – XV 64 – 15

Блок-схема адгоритму для десаткової системи числення.



Часова складність L = 7
Програмна складність P =7

Блок-схема адгоритму для римської системи числення.



Еквівалентні перетворення здійснюються за допомогою скінченої таблиці еквівалентних перетворювань .

II=I+I; III=II+I; IV=III+I; V=IV+I; VI=V+I; VII=V+II; VIII=V+III; IX=VIII+I; X=IX+I;.....L=XL+X; ... M=CM+C;

Часова складність L = 10
Програмна складність P =10