**Практична робота 1.**

**Тема**: Поняття алгоритму. Властивості алгоритмів. Форми подання алгоритмів.

Виконавець алгоритму.

**Мета:**

*навчальна*:сформувати  поняття алгоритму, форм подання алгоритму, виконавця алгоритму; сформувати уявлення про: різні види алгоритмів та їх виконання, різні методи подання алгоритмів, загальні основи термінології алгоритмізації; формувати вміння використовувати: графічні схеми базових структур, алгоритмів для опису алгоритмів, різні засоби подання алгоритмів.

*розвивальна*: розвивати логічне мислення, розумову активність учнів, цілеспрямованість, пам'ять;

*виховна*: уважність, вміння самостійно приймати рішення.

**Тип уроку**: засвоєння нових знань, формування вмінь.

1. **Алгоритм. Історична довідка.**

Слово алгоритм походить від імені  перського вченого, астронома,  математика Аль-Хорезмі IX століття.

В першій половині [XII століття](http://uk.wikipedia.org/wiki/12_%D1%81%D1%82%D0%BE%D0%BB%D1%96%D1%82%D1%82%D1%8F) його книга потрапила до [Європи](http://uk.wikipedia.org/wiki/%D0%84%D0%B2%D1%80%D0%BE%D0%BF%D0%B0) . Переклад  [латинською мовою](http://uk.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B8%D0%BD%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) був таким *Algoritmi de numero Indorum*. Вважається, що перше слово в перекладі відповідає  невдалій латинізації імені Аль-Хорезмі, а назва перекладу звучить як «Алгорітми про індійську лічбу». Це зрозуміле й точне розпорядження виконати  послідовність дій, спрямованих на досягнення зазначеної мети.

Перший алгоритм, призначений для виконання на автоматичному  обчислювальному пристрої ([комп'ютері](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80)), описала [Ада Лавлейс](http://uk.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0_%D0%9B%D0%B0%D0%B2%D0%BB%D0%B5%D0%B9%D1%81) в [1843](http://uk.wikipedia.org/wiki/1843) році. Алгоритм мав обчислювати [числа Бернуллі](http://uk.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B0_%D0%91%D0%B5%D1%80%D0%BD%D1%83%D0%BB%D0%BB%D1%96) й працювати на [аналітичній машині Беббіджа](http://uk.wikipedia.org/w/index.php?title=%D0%90%D0%BD%D0%B0%D0%BB%D1%96%D1%82%D0%B8%D1%87%D0%BD%D0%B0_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%91%D0%B5%D0%B1%D0%B1%D1%96%D0%B4%D0%B6%D0%B0&action=edit&redlink=1). Цей алгоритм вважається першою [комп'ютерною програмою](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), а його розробниця, Ада Лавлейс — першим програмістом.

***Алгоритм -*** це точний і зрозумілий опис послідовності дій над заданими об’єктами , що дозволяє отримати кінцевий результат. До слова алгоритм близькі за значенням слова спосіб, рецепт. Найбільше прикладів алгоритмів в математиці – науці, у якій власне й зародилося це поняття. По суті, математика вивчає різні алгоритми і створює нові. До алгоритмів належать правила виконання арифметичних дій.

Однак алгоритми в інформатиці – це не тільки рецепт розв’язування задач. Алгоритми розробляють, насамперед з метою автоматизації дій виконавця.

Складання алгоритму починається з розбивання описуваного процесу на послідовність окремих кроків. Властивість розбивання алгоритму на окремі кроки називають дискретністю алгоритму. Кожний крок алгоритму формулюється в вигляді інструкцій (команд), тобто визначених дій виконавця. Для вас алгоритм – це складання технологічної карти.

**2. Виконавець алгоритму.** **Основні характеристики.**

Алгоритми складаються з орієнтацією на певного виконавця алгоритму.

Виконавець алгоритму – це істота (людина, свійська тварина) або неістота (робот, автомат , комп’ютерна система), яка може виконувати всі вказівки заданого алгоритму.

Основні характеристики виконавця алгоритму:

1. Середовище виконання – умови, в яких може діяти виконавець.
2. Елементарні дії – найпростіші дії, які може виконати виконавець.
3. Система команд виконавця – сукупність допустимих команд виконавця.

Допустимі команди – команди, які зрозумілі виконавцю і можуть бути ним виконані. Недопустимі команди – команди, які не можуть бути виконані виконавцем.

Процес алгоритмізації – це визначення елементарних дій і порядку у вигляді послідовності команд виконання.

Отже, від виконавця алгоритму залежить алгоритм розв’язування однієї й тієї самої задачі. Наприклад: риття ями (виконавці — людина або екскаватор), розв'язування математичної задачі (учень або комп'ютер). Розглянемо декілька прикладів, пов’язаних з вашою професією. Потрібно приготувати м’ясний фарш. Виконавцем алгоритму можете бути ви, працюючі на звичайній м’ясорубці або електром’ясорубці.

Усі ми з вами любимо смачні справи, наприклад домашню локшину, чебуреки, а також тістечка з листкового або пісочного тіста. Готуючи ці страви, доводиться докласти чимало зусиль, щоб тісто було тонким. Вдома ми використовуємо звичайну качалку, але в умовах виробництва іі використовувати недоцільно. Щоб розкачати пласт тіста, на підприємствах харчування використовують машину для розкачування тіста МРТ-60М. Виконавцем даного процесу можете бути кухар або тісторозкачувальна машина, яка призначена для розкачування крутого дріжджового, листкового і пісочного тіста.

Отже, поняття алго­ритму в інформатиці є фундаментальним, тобто та­ким, яке не визначається через інші ще більш прості поняття (для порівняння у фізиці — поняття простору й часу, у математиці — точка).

Поняття алгоритму належить до фундаментальних понять математики. Доречним прикладом із математики можна навести відому теорему Піфагора: «Квадрат гіпотенузи дорівнює сумі квадратів катетів». Керуючись тільки формулюванням цієї тео­реми, ви можете дістати потрібний результат, тобто гіпотенузу.

1. **Властивості алгоритмів**

Зрозумілість.

Визначеність.

Дискретність

Масовість

Результативність.

Формальність.

Скінченність.

1. **Зрозумілість**

Щоб виконавець міг досягти поставленої мети, користуючись алгоритмом, він повинен уміти виконувати кожну його вказівку, тобто розуміти кожну з команд, що входять до алгоритму.

Наприклад, вам потрібно приготувати стейк. Якщо вам це завдання дали б на початку першого курсу, то ви не знали як правильно виконати це завдання. Зараз на другому курсі, ви знаєте яку частину нарізки вам потрібно брати, знаєте час приготування різних видів стейків за часом просмаження.

1. **Результативність.**

Очевидно, що виконання будь-якого алгоритму повинне завершуватися одержанням результатів. Тобто ситуації, що в деяких випадках можуть призвести до так званого «зациклення», повинні бути виключені під час складання алгоритму. Наприклад, розглянемо таку ситуацію: роботові дано завдання залишити кімнату (замкнутий простір), не виконуючи руйнівних дій. У цьому випадку, якщо роботу не дати вказівки відчинити двері (що, можливо, зачинені), то спроби залишити приміщення можуть бути безуспішними.

Наприклад, вам потрібно приготувати яйця. «Мішечок», рідкі та круті. Ви маєте одну сировина, один спосіб теплової обробки, але різний результат в залежності від часу варіння. Яйця «мішечок» готуються - 4-5 хв; рідкі - 2 -3хв.; круті – 7-8 хв.

1. **Визначеність (однозначність, детермінованість)**

Зрозумілий алгоритм все ж таки не повинен містити вказівки, зміст яких може сприйматися неоднозначне. Наприклад, вказівки «почисти картоплю», «посоли за смаком», «взяти 2-3 ложки цукру», «трохи підігрій молоко», «прибери в квартирі» є неоднозначними, тому що в різних випадках можуть призвести до різних результатів. Не випадкову для приготування страви за рецептом вказується конкретна маса продуктів, що використовуються.

Наприклад:  рецепт сирників: 250 г сиру домашнього; 1 велике яйце; 2 ст. л. цукру; 1 ч. л. ванільного цукру; 2 ст. л. з гіркою борошна.

Точність — це властивість алгоритму, що полягає в тому, що алгоритм повинен бути однозначно витлумачений і на кожному кроці виконавець повинен знати, що йому робити далі.

1. **Дискретність(розривність)**

Дискретність алгоритму означає, що його виконання зводиться до виконання окремих дій (кроків) у певній послідовності.

При цьому для виконання дій їх розбивають у визначеній послідовності на прості кроки. Виконати дії наступного розпорядження можна лише виконавши дії попереднього. Наприклад, виготовлення сирників за запропонованим рецептом: Змішати до однорідної маси всі інгредієнти. Сформувати 7 кульок, злегка запанірувати їх у борошні, розплескати і обсмажити на олії.

1. **Масовість**

Масовість алгоритму означає,що алгоритм може бути застосованим до цілого класу однотипних задач, для яких спільними є умова та хід розв’язання.

Дуже важливо, щоб складений алгоритм забезпечу­вав розв'язання не однієї окремої задачі, а міг вико­нувати розв'язання широкого класу задач цього типу. Наприклад, алгоритм розв’язування квадратного рівняння. Отже, під масовістю ал­горитму мається на увазі можливість його застосу­вання для виконання великої кількості однотипних завдань.

Наприклад випікання смаколиків виробів кондитерським цехом.

Однак поряд з масовими алгоритмами складаються і застосовуються немасові алгоритми. Наприклад, алгоритм приготування конкретного салату (наприклад, грецького) на конкретну кількість осіб.

1. **Формальність.**

Виконання алгоритмів носить формальний характер: виконавець не повинен розуміти жодну з команд, може не знати мети виконання алгоритму і все одно отримає результат. На цьому побудовані всі технічні пристрої, призначені для виконання різноманітних алгоритмів.

Картопляне пюре протирання в гарячому стані, щоб воно було пишне та однорідне. Або випікання коржів торта «Наполеон» в духовці при температурі 220оС.

1. **Скінченність.**

Скінченність алгоритму означає, що його виконання виконавець закінчить після скінченної (можливо, досить великої) кількості кроків і за скінчений час при будь-яких допустимих значеннях початкових даних. Продовжимо наш приклад: випікання коржів торта «Наполеон» при температурі 220 о С за 7-8 хв.

**4. Форми подання алгоритму**

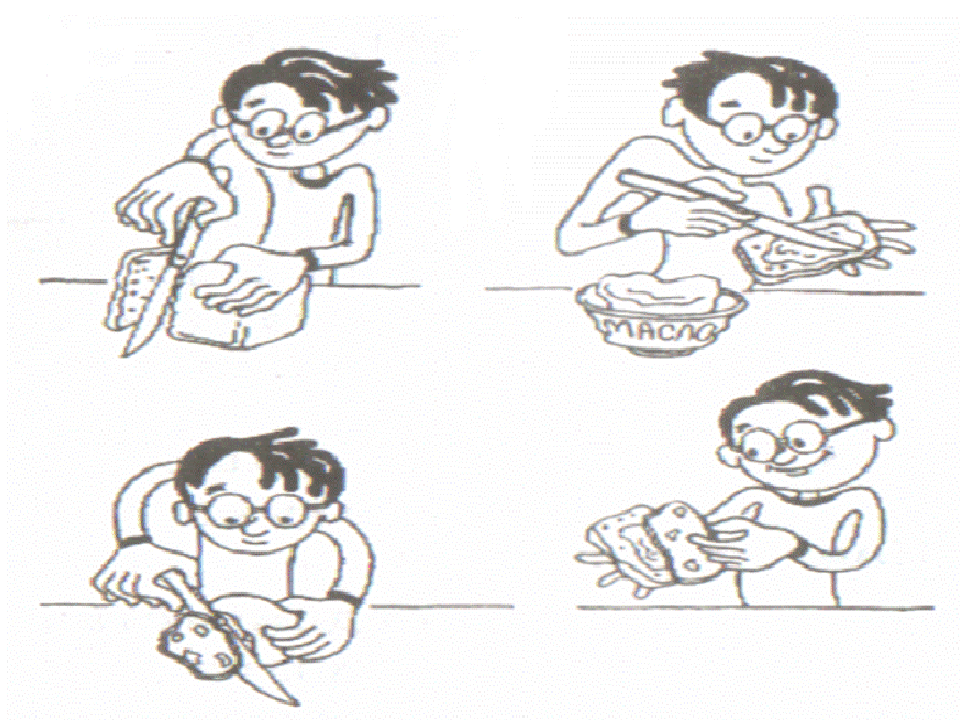
*Перший спосіб — словесний (кулінарні рецепти, правила).*

Подання алгоритмів виконавцю за допомогою сло­весного опису.

Приклад №1. Любий кулінарний рецепт.

*Другий спосіб — словесно-формульний (*подача у вигляді *таблиць, формул, схем, малюнків)*.

Приклад №1. Приготування бутерброду з маслом та сиром.



Приклад №2. Правила роботи учнів в комп’ютерному класі.

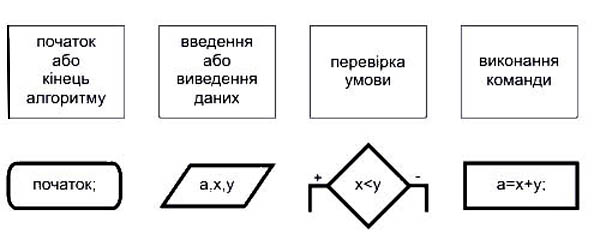
Приклад №3. Алгоритмів, що записані у вигляді умовних позначок на купленому товарі, щодо його користування (заварювання чаю, приготування різних страв, використовуючи інструкцію на упаковці товару, прання білизни тощо).

Приклад №4. У математиці наявність формул дозволяє розв'язати задачу навіть «не використовуючи слів».

*Третій спосіб — графічний (*запис за допомогою *блок-схеми).*

Цей метод був запропонований в інформатиці для наочності представлення алгоритму за допомогою набору спеціальних блоків, для полегшення переходу від запису алгоритмів до написання програм.

*Основні елементи блок - схем алгоритму.*



*Четвертий спосіб* — *навчальні алгоритмічні мови* (псевдокоди).

Ці мови мають жорстко визначений синтаксис і вже максимально наближені до машинної мови (мови програмування). Але створені вони з навчальною метою, тому мають зрозумілий для людей вигляд, Ці псевдокоди мають програмну реалізацію і дуже широко застосовуються на етапі навчання основ програмування.

*П'ятий спосіб* максимально наближений до комп'ютера — це *мови програмування.*

Справа в тому, що найчастіше в практиці виконавцем створеного людиною алгоритму є машина, тож алгоритм повинен складатися мовою, зрозумілою для комп'ютера, тобто мовою програмування.

*Інші способи* – специфічні способи подання алгоритмів, такі, як запис музики за допомогою нот.

5. **Графічне зображення базових  алгоритмічних структур**.

|  |  |
| --- | --- |
| Назва блоку | Опис дії |
|  | Позначає початок та кінець алгоритму |
|  | Позначає ввід вихідної інформації і вивід проміжної чи результуючої інформації |
|  | Позначає дію, яку треба виконати |
|  | Позначає перевірку значення логічного виразу деякої умови |

*Просте слідування*

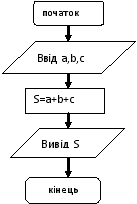
Слідування означає, що дії повинні виконуватись послідовно одна за одною.



Лінійний алгоритм – алгоритм, в якому всі вказівки виконуються одна за одною і не містить розгалужень та повторень.

**Приклад**

Алгоритм знаходження суми S трьох чисел a,b,c.



*Розгалуження*

Розгалуження – це така форма організації дій, які містять умови і в залежності від того чи вона виконується чи ні здійснюється або одна або друга послідовність дій.

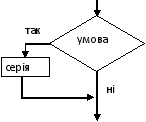
Умова - це будь-яке твердження, яке або виконується або не виконується, тобто можна дістати одну з двох відповідей: «так» або «ні». Блок-схемою це можна зобразити так:



Якщо умова виконується, то виконується серія команд 1 (гілка так), якщо умова не виконується, то виконується команд 2 (гілка ні). Після виконання серії команд виконавець переходить до наступної команди після команди розгалуження.

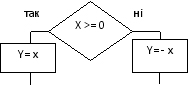
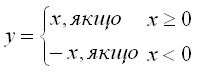
Можливий випадок, що у випадку невиконання умови не потрібно виконувати ніяких дій. Тоді використовується скорочена форма розгалуження («Якщо-то»).

Блок-схема структури «якщо-то»

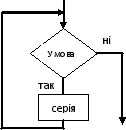
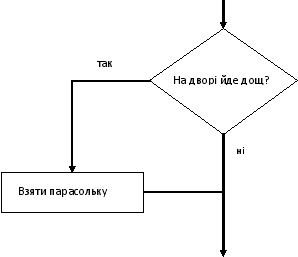


Приклад

Алгоритм обчислення модуля:



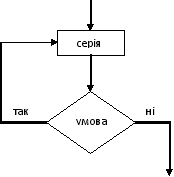
Приклад



*Повторення (цикл)*

Часто зустрічаються такі задачі при виконанні яких потрібно виконувати одні і ті самі дії декілька разів. Тоді кажуть, що така структура команд називається циклічною, або утворена структура «повторення».

Цикл – це форма організації дій, за якою одна і та сама послідовність дій виконується кілька разів доти, поки виконується деяка умова. Серія команд, що виконується декілька разів без змін при кожному проході циклу, називається тілом циклу.



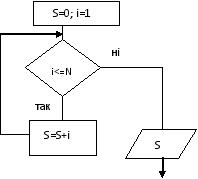
Є два типи повторень: з передумовою та післяумовою. У першому випадку спочатку перевіряється умова і, якщо вона істинна, то вказана дія виконується черговий раз, якщо ж ні – то виконання дії припиняється.

У випадку повторення з післяумовою спочатку виконується серія команд, а після цього перевіряється умова і визначається, чи є потреба виконувати її знову.

Можливі ситуації, коли «цикл поки» не виконується жодного разу. Це відбувається в тому випадку, коли на першому кроці умова є хибною. Якщо при повторенні циклу умова залишається завжди  істинною, то цикл може повторюватись нескінченно.

Приклад

Алгоритм підрахунку суми N перших натуральних чисел. Суму позначимо через S, через і – черговий доданок. Спочатку S=0, оскільки ще суми не знаходили, i=1 (перше натуральне число). Щоб знайти суму, то потрібно до попередньої суми додати наступний доданок: S=S+i. Для отримання наступного числа потрібно попереднє збільшити на одиницю: i=i+1. Виконання циклу продовжується до тих пір, поки i<=N.



**Принципи структурного програмування**

Алгоритми, у яких використовується тільки структура «слідування», називаються  лінійними.

Алгоритми, в основі яких лежить структура «розгалуження», називаються алгоритмами з розгалуженнями.

Алгоритми, в основі яких лежить структура «повторення», називають циклічними.

На практиці алгоритми розв’язування складних задач містять у собі всі три типи базових структур алгоритмів. Розглянуті принципи конструювання алгоритмів називають принципами структурного програмування.