**Лекція №4. Побудова і аналіз алгоритмів. Методи розробки алгоритмів.**

Питання студентам для контролю знань попередньої лекції і матеріалу для самостійного вивчення.

1. Надайте визначення інформації та даних.
2. Як пов’язані інформація і дані?
3. Поняття структури даних. Як вона характеризується?
4. Класифікація структур даних у програмах користувача.
5. Визначте рівні подання структур даних.
6. Назвіть елементи логічної структури економічної інформації.
7. Надайте визначення інформаційного масиву даних та інформаційного потоку.
8. Окресліть взаємозв'язок між елементами логічної структури економічної інформації.
9. Визначте одиниці фізичної структури даних.
10. Наведіть приклади лінійних та нелінійних структур даних.
11. Чим відрізняється стек від черги?
12. Назвіть складені типи даних.

**Формалізація алгоритмів**.

Процес створення комп’ютерної програми для вирішення будь-якої практичної задачі складається з декількох етапів:

* формалізація і створення технічного завдання на вихідну задачу;
* розробка алгоритму вирішення задачі;
* написання, тестування, наладка і документування програми;
* отримання розв’язку вихідної задачі шляхом виконання програми.

Практично будь-яку галузь математики або інших наук можна застосувати до побудови моделі певного класу задач. Для задач, числових за своєю природою, можна побудувати моделі на основі загальних математичних конструкцій, таких як системи лінійних рівнянь, диференціальні рівняння. Для задач з символьними або текстовими даними можна застосувати моделі символьних послідовностей або формальних граматик. Вирішення таких задач містить етапи компіляції і інформаційного пошуку.

Коли побудована чи підібрана потрібна модель вихідної задачі, то природно шукати її вирішення в термінах цієї моделі. На цьому етапі основна мета полягає в побудові розв’язку в формі алгоритму, який складається з скінченої послідовності інструкцій, кожна з яких має чіткий зміст і може бути виконана з скінченими обчислювальними затратами за скінчений час. Інструкції можуть виконуватися в алгоритмі будь-яку кількість раз, при цьому вони самі визначають цю кількість повторень. Проте вимагається, щоб при будь-яких вхідних даних алгоритм завершився після виконання скінченої кількості інструкцій. Таким чином, програма, яка написана на основі розробленого алгоритму, при будь-яких початкових даних ніколи не повинна приводити до нескінченних циклічних обчислень.

**Покрокове проектування алгоритмів**

В основу процесу проектування програми з розбиттям її на достатню кількість дрібних кроків можна покласти наступну послідовність дій:

1. Вихідним станом процесу проектування є більш-менш точне формулювання мети алгоритму, або конкретніше – результату, який повинен бути отриманий при його виконанні. Формулювання проводиться на природній мові.
2. Проводиться збір фактів, які стосуються будь-яких характеристик алгоритму, і робиться спроба їх представлення засобами мови.
3. Створюється образна модель процесу, який відбувається, використовуються графічні та інші способи представлення, схеми, які дозволяють краще зрозуміти виконання алгоритму в динаміці.
4. В образній моделі виділяється найбільш суттєва частина, для якої підбирається найбільш точне словесне формулювання.
5. Проводиться визначення змінних, які необхідні для формального представлення даного кроку алгоритму.
6. Вибирається одна з конструкцій – проста послідовність дій, умовна конструкція або циклу. Складні частини вибраної формальної конструкції (умова, заголовок циклу) залишаються в словесному формулюванні.
7. Для інших неформалізованих частин алгоритму, які залишились у словесному формулюванні – перерахована послідовність дій повторюється.

Схематично узагальнений процес програмування можна представити наступною схемою.



На першому етапі створюється модель вихідної задачі, для чого застосовуються відповідні математичні моделі. На цьому етапі для знаходження рішення також будується неформальний алгоритм.

На наступному етапі алгоритм записується на псевдомові – композиції конструкцій мови програмування і менш формальних і узагальнених операторів на простій мові. Продовженням цього етапу є заміна неформальних операторів послідовністю більш детальних і формальних операторів. З цієї точки зору програма на псевдомові повинна бути достатньо детальною, так як в ній фіксуються (визначаються) різні типи даних, над якими виконуються оператори. Потім створюються абстрактні типи даних для кожного зафіксованого типу даних (за винятком елементарних типів даних, таких як цілі й дійсні числа або символьні стрічки) шляхом завдання імен функцій для кожного оператора, який працює з даними абстрактного типа, і заміни їх (операторів) викликом відповідних функцій.

Третій етап процесу – програмування – забезпечує реалізацію кожного абстрактного типа даних і створення функцій для виконання різних операторів над даними цих типів. На цьому етапі також замінюються всі неформальні оператори псевдомови на код мови програмування. Результатом цього етапу повинна бути виконувана програма. Після її наладки отримують працюючу програму.

Алгоритм повинен мати наступні п’ять якостей:

1. Обмеженість в часі – робота алгоритму обов’язково повинна завершитись через деякий розумний період часу.
2. Правильність – алгоритм повинен знаходити правильне, а не будь-яке рішення.
3. Детермінованість – скільки б разів не виконувався алгоритм з однаковими вхідними даними, результат повинен бути однаковим.
4. Скінченність – опис роботи алгоритму повинен мати скінчену кількість кроків.
5. Однозначність – кожний крок алгоритму повинен інтерпретуватися однозначно.

Алгоритм повинен задовольняти наступні протиріччя.

1. Бути простим для розуміння, перекладу в програмний код і наладки.
2. Ефективно використовувати комп’ютерні ресурси і виконуватися швидко.

Якщо написана програма повинна виконуватися лише декілька разів, то перша вимога найбільш важлива. Вартість робочого часу програміста, звичайно, значно перевищує вартість машинного часу виконання програми, тому вартість програми оптимізується за вартістю написання (а не виконання) програми. Якщо мати справу з задачею, вирішення якої потребує значних обчислювальних затрат, то вартість виконання програми може перевищити вартість написання програми, особливо якщо програма повинна виконуватися багаторазово. Тому, з економічної точки зору, перевагу буде мати складний комплексний алгоритм (в надії, що результуюча програма буде виконуватися суттєво швидше, ніж більш проста програма). Але і в цій ситуації розумніше спочатку реалізувати простий алгоритм, щоб визначити, як повинна себе вести більш складна програма. При побудові складної програмної системи бажано реалізувати її простий прототип, на якому можна провести необхідні виміри й змоделювати її поведінку в цілому, перш ніж приступати до розробки кінцевого варіанту. Таким чином, програмісти повинні бути обізнані не тільки з методами побудови швидких алгоритмів, але й знати, коли їх потрібно застосувати.

Важливо розрізняти практичну складність, яка є точною мірою часу обчислення і об’єму пам’яті для конкретної моделі обчислювальної машини, і теоретичну складність, яка більш незалежна від практичних умов виконання алгоритму і дає порядок величини вартості.

Більшість алгоритмів надає вибір між швидкістю виконання і ресурсами. Задача може виконуватися швидше, використовуючи більше пам’яті, або навпаки – повільніше з меншим обсягом пам’яті.

**Методи розробки алгоритмів**

При розробці алгоритма (програми) застосовуються різні методи. Найпоширенішими є методи:

*зверху-вниз* – складна проблема розкладається поступово на простіші;

*знизу-вверх* – виділяються простіші елементи, які  об’єднуються поступово у складніші;

*розширення ядра* – спочатку виділяється основна (головна) частина алгоритму (скелет алгоритму), яка поступово доповнюється рештою елементів.

Декомпозиція при програмуванні має на меті розкладання складної проблеми (задачі) на простіші, інтеграція – реалізацію складної проблеми (задачі) вищого рівня за допомогою проблем (задач) нижчого рівня. Кроків декомпозиції може бути декілька, тобто проблема (задача) деякого рівня декомпозиції може підлягати подальшій декомпозиції. Цей процес породжує ієрархію проблем (задач, виконавців).

Декомпозиція при розробці програми породжує ієрархічну структуру програми. Ієрархічній структурі програми відповідає ієрархія абстрактних виконавців – ієрархія дій, даних, керування. Для кожного абстрактного виконавця ієрархії будується алгоритм відповідного рівня абстракції. При побудові такого алгоритма треба використовувати структури керування й структри абстрактних даних. При розробці програми для структур керування й структур даних використовуються певні засоби зображення.

**Низхідне проектування**

*Метод низхідного проектування* припускає послідовне розкладання загальної функції обробки даних на прості функціональні елементи ("зверху-вниз").

В результаті будується ієрархічна схема, що відображає склад і взаємозалежність окремих функцій, яка носить назву *функціональна структура алгоритму (ФСА)* застосування.

Послідовність дій з розробки функціональної структури алгоритму застосування:

* визначаються цілі автоматизації наочної області і їх ієрархія *(мета-підціль);*
* встановлюється склад застосувань (завдань обробки), що забезпечують реалізацію поставлених цілей;
* уточнюється характер взаємозв'язку застосувань і їх основні характеристики (інформація для вирішення завдань, час і періодичність рішення, умови виконання і ін.);
* визначаються необхідні для вирішення завдань функції обробки даних;
* виконується декомпозиція функцій обробки до необхідної структурної складності, що реалізовується передбачуваним інструментарієм.

Розкладання повинне носити строго *функціональний* характер, тобто окремий елемент ФСА описує *закінчену змістовну функцію* обробки інформації, яка припускає певний спосіб реалізації на програмному рівні.

Функції введення-виводу інформації рекомендується відокремлювати від функцій обчислювальної або логічної обробки даних.

По частоті використання функції діляться на:

* одноразово виконувані;
* що повторюються.

Ступінь деталізації функцій може бути різною, але ієрархічна схема повинна давати уявлення про склад і структуру взаємозв'язаних функцій і загальний алгоритм обробки даних. Широко використовувані функції набувають рангу стандартних (вбудованих) функцій при проектуванні внутрішньої структури програмного продукту.

**Висхідний метод**, навпаки, спираючись на деякий, заздалегідь визначуваний коректний набір підалгоритмів, будує функціонально завершені підзадачі більш загального призначення, від них переходить до більш загальним, і так далі, до тих пір, поки не дійдемо до рівня, на якому можна записати рішення поставленої задачі. Цей метод відомий як метод проектування "від низу до верху".

Побудова алгоритму для розв’язування завдання з будь-якої галузі потребує від людини глибоких знань у цій галузі, буває пов’язана з ретельним аналізом поставленого завдання, складними, іноді громіздкими міркуваннями. На пошуки алгоритму розв’язання деяких завдань вчені витравчають дуже багато років.

Але коли алгоритм створено, то розв’язання завдання за готовим алгоритмом уже не потребує будь-яких міркувань і зводиться лише до чіткого виконання команд алгоритму. Найпростіші операції, на які при створенні алгоритму розбивається весь процес розв’язування завдання, може реалізувати і машина, спеціально створена для виконання команд цього алгоритму, якщо буде виконувати їх у послідовності, що зазначена в алгоритмі. На цьому положенні й грунтується робота автоматичних пристроїв.

Алгоритми є основою комп'ютерних наук і програмування. Реальна продуктивність програмної системи визначається рівно двома факторами:

*(1) обраними для неї алгоритмами і  
(2) адекватністю і ефективністю різних рівнів реалізації.*

Тому хороше проектування алгоритмів є вирішальним для продуктивності будь-яких програмних систем. Більш того, вивчення алгоритмів забезпечує проникнення в глибинну природу розв'язуваної задачі і вибір методів її вирішення, незалежних від мови програмування, парадигми програмування, архітектури комп'ютера і інших аспектів реалізації.

Важливою частиною обчислювальної науки є вміння вибирати і застосовувати алгоритми, що відповідають конкретним цілям, а також враховувати можливість того, що таких алгоритмів може не існувати. Це вміння грунтується на вивченні алгоритмів, використовуваних для вирішення відомого кола чітко визначених завдань, усвідомленні їх сили і слабкості, а також застосування в конкретних контекстах. Лейтмотивом даної сфери знань є ефективність.

Аналіз алгоритму полягає в тому, щоб передбачити необхідні для його виконання ресурси. Іноді оцінюється потреба в таких ресурсах, як пам'ять, пропускна здатність мережі або необхідне апаратне забезпечення, але частіше за все визначається час обчислення. Шляхом аналізу декількох алгоритмів, призначених для вирішення однієї і тієї ж задачі, можна без напруги вибрати найбільш ефективний. У процесі такого аналізу може також опинитися, що кілька алгоритмів приблизно рівноцінні, а все решта потрібно відкинути.

Завдання, що підлягає вирішенню на ЕОМ, може бути охарактеризована кількістю даних, складністю алгоритму, трудомісткістю алгоритму.

Під складністю алгоритму розуміється кількість інформації, необхідної для його опису. Оцінка складності алгоритму може бути дана в бітах, байтах, кількості символів певної мови. Для оцінки можуть використовуватися також число операторів конкретного алгоритмічної мови, кількість машинних кодів і т.д. Попередня оцінка складності алгоритму і кількості даних виконується експертним шляхом, а точні значення можуть бути відомі тільки після завершення програмування. Складність алгоритму і кількість даних характеризують потребу завдання в оперативної і зовнішньої пам'яті.

Під трудомісткістю алгоритму можна розуміти кількість обчислювальної роботи, необхідної для його реалізації. Трудомісткість характеризує витрати часу для реалізації алгоритму на деякій сукупності технічних засобів. Зазвичай трудомісткість оцінюється кількістю процесорних операцій і операцій введення-виведення.

Алгоритми, як і апаратне забезпечення комп'ютера, являють собою технологію. Загальна продуктивність системи настільки ж залежить від ефективності алгоритму, як і від потужності застосовного апаратного забезпечення. В області розробки алгоритмів відбувається такий же швидкий розвиток, як і в інших комп'ютерних технологіях.

**Створення алгоритму**

Будь-який алгоритм потрібно розглядати як опис процедури, яка обробляє вхідні дані й отримує вихідні дані. Вхідні дані ще називають входом алгоритму, або його аргументами, а вихідні дані - виходом алгоритму, або результатом.

Алгоритм вважається правильним, якщо при будь-якому допустимому для даної задачі наборі вхідних даних він завершує свою роботу і отримує результат, що задовольняє вимоги задачі. У цьому випадку кажуть, *що* алгоритм розв'язує дану задачу. Неправильний алгоритм може або не зупинитися, або дати неправильний результат.

Усі алгоритми поділяються на *два великі класи.* Це ***алгоритми з повторениями***та ***рекурсивні алгоритми****.*

В основі **алгоритмів з повторениями** лежать операції переходу та умовні вирази, аналіз яких і виконує відповідні переходи. Для аналізу таких алгоритмів треба оцінити кількість операцій усередині циклу та кількість повторень цих циклів.

**Рекурсивні алгоритми** поділяють велику задачу на підзадачі, до кожної з яких окремо знову застосовуються ці самі алгоритми. Такі алгоритми ще відносять до алгоритмів, основою створення яких є парадігма «*розділяй і володарюй*». Перевагою таких алгоритмів є невеликий, простий і зрозумілий запис. Аналіз рекурсивних алгоритмів вимагає підрахунку кількості операций, необхідних для поділу задачі на підзадачі, виконання алгоритму в кожній із них та об'єднання отриманих результатів для розв'язування задачі в цілому.

**Парадигма** - набір теорій, стандартів і методів, які спільно представляють собою спосіб організації наукового знання, іншими словами, спосіб бачення світу. За аналогією з цим прийнято вважати, що парадигма в програмуванні - спосіб концептуалізації, який визначає, як слід проводити обчислення, і як робота, виконувана комп'ютером, повинна бути структурована і організована

Розділяй і володарюй (англ. Divide and conquer) - важлива парадигма розробки алгоритмів, що полягає в рекурсивному розбитті розв'язуваної задачі на дві або більше підзадач того ж типу, але меншого розміру, і комбінуванні їх рішень для отримання відповіді до вихідного завдання; розбиття виконуються до тих пір, поки всі підзадачі не опиняться елементарними.

У цьому визначенні слід пояснити поняття рекурсії. Термін «рекурсія» використовується в різних спеціальних галузях знань, але найбільш ефективно в програмуванні. Найпростіший варіант побачити рекурсію - це навести Web-камеру на екран монітора комп'ютера, природно, попередньо її включивши. Таким чином, камера буде записувати зображення екрану комп'ютера, і виводити його ж на цей екран, вийде щось на зразок замкнутого циклу. У підсумку ми спостерігатимемо щось схоже на тунель.

Рекурсія - це визначення об'єкта через самого себе, тобто якщо ми рекурсивно визначаємо функцію, то вона викликає саму себе, безпосередньо (в своєму тілі) або опосередковано (через іншу функцію).

Отже, створення алгоритму - це один з етапів розв'язування поставленої задачі. Але він не відокремлений від усіх інших етапів, таких як: визначення моделі задачі, що розв'язується; запис алгоритму мовою програмування; тестування програми на комп'ютері; отримання розв'язку поставленої задачі шляхом виконання завершеної програми.

**Структурні принципи алгоритмізації**

Структурні принципи алгоритмізації (структурні методи алгоритмізації) – це принципи формування алгоритмів з базових структурних алгоритмічних одиниць (проходження/слідування, розгалуження, повторення), використовуючи їх послідовне з'єднання або вкладення один в одного з дотриманням певних правил, що гарантують читабельність і виконуваність алгоритму зверху вниз і послідовно. *.* В 1966 році вчені Бром і Джакопіні довели наступну теорему: «Кожен алгоритм може бути побудований за допомогою лише трьох базових алгоритмічних структур – слідування, розгалуження, циклу». Це **Теорема про структуризацію:** будь-яка проста програма може бути перетворена у функціонально їй еквівалентну програму, побудовану на основі 3 наступних структур:

*Базові структури алгоритмів*

1. Проходження / слідування.
2. Якщо-то-інакше (розгалуження).
3. Цикл з передумовою.

Структура слідування, або лінійна, містить вказівки, що виконуються послідовно одна за одною.

Розгалужена структура передбачає вибір дії залежно від дотримання деякої умови, при цьому деякі вказівки можуть не виконуватися взагалі.

Умови, що використовуються в розгалуженому алгоритмі поділяються на прості і складені. Прості умови записують за допомогою двох виразів поєднаних знаками відношень >, <, >=, <=, =, <>. Результатом перевірки умови є логічний вираз ІСТИНА (TRUE), якщо умова виконується, або ХИБНІСТЬ (FALSE), якщо умова не виконується.

Складена умова містить дві і більше простих умов, поєднаних знаками логічних операцій:

* + i, &, and, наприклад - <умова1> & <умова2>, and(умова1;умова2) – результат буде істинним тоді і тільки тоді, коли обидві прості умови будуть дотримані.
  + або, or, наприклад - <умова1> or <умова2>, or(умова1;умова2) – результат буде істинним тоді, коли хоча б одна з умов буде дотримана.
  + не, not, наприклад – not(умова1) – це є заперечення умови1, результатом буде твердження, протилежне умові1.

Таблиця 1.1.2

**Таблиця істинності тверджень**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Умова1 | Умова2 | and(Умова1;Умова2) | Or(Умова1;Умова2) | Not(Умова1) |
| T | T | T | T | F |
| F | T | F | T | T |
| T | F | F | T | F |
| F | F | F | F | T |

Циклічна структура використовується при необхідності повторень виконання деяких дій з різними (новими) значеннями вхідних даних, тобто коли треба виконати перебір варіантів. Алгоритмі циклічної структури описують цикли трьох видів: безумовний або арифметичний цикл (цикл ДЛЯ); умовний цикл з передумовою (цикл ПОКИ) та умовний цикл з післяумовою (цикл ДО).

Як правило, умовні цикли використовують тоді, коли невідома кількість повторень вказівок циклу, такі цикли ще називають ітераційними.

**Цілі структурного програмування**

1. Забезпечити дисципліну програмування
2. Поліпшити читабельність програми
3. Підвищити ефективність програми

**Тестування алгоритму**

**Тестування алгоритму** – це перевірка правильності або неправильності роботи алгоритму на спеціально заданих тестах або тестових прикладах – завданнях з відомими вхідними даними і результатами (іноді достатні їх наближення). Тестовий набір повинен бути мінімальним і повним, тобто що забезпечує перевірку кожного окремого типу наборів вхідних даних, особливо виняткових випадків.

Тестування алгоритму не може дати повної (100%-ой) гарантії правильності алгоритму для всіх можливих наборів вхідних даних, особливо для достатньо складних алгоритмів.

Для економістів інтерес представляє саме економічна інформація.

Економічну інформацію можна розглядати як ресурс, аналогічний матеріальним, трудовим і грошовим ресурсам.

Інформаційні ресурси - сукупність накопиченої інформації, зафіксованої на матеріальних носіях в будь-якій формі, що забезпечує її передачу в часі і просторі для вирішення наукових, виробничих, управлінських і інших завдань.

Під економічною інформацією розуміють сукупність даних (відомостей), що їх використовують для здійснення функцій управління народним господарством і його окремими ланками. До найважливіших властивостей економічної інформації належать:

* переважання алфавітно-цифрової форми подання;
* висока питома вага вихідних даних, що підлягають обробці;
* значний обсяг умовно-сталої інформації;
* дослідження таких інформаційних одиниць, як обліково-планові показники;
* висока точність результатів розв’язання задач;
* широке використання документної форми носіїв даних;
* переважання дискретної форми числових величин.

Крім загальних властивостей, кожний різновид економічної інформації характеризується властивостями, що випливають із методологістичних принципів розв’язання тих чи інших функціо­нальних завдань управління.

На вдосконалення управлінської діяльності істотно впливають раціоналізація потоків інформації, процесів руху її та перетворення; організація систем інформації; поліпшення забезпечення інформацією керуючих елементів. Тому питання управлінської й інформаційної діяльності тісно взаємозв’язані.

Термін «інформація» означає повідомлення, відомості, знання, потрібні для прийняття рішень. Поняття якості інформації невіддільне від процедур прийняття рішень: тільки цілком вірогідна інформація може забезпечити якість рішення, допомогти керівникові уникнути помилок. Інформатика виступає тут як дисципліна, що створює технологію роботи з інформацією. Справді, брак інформації призводить до того, що особа, яка приймає рішення, легко може припуститися помилки. Недоліки планування часто-густо спричинені браком надійної інформації. Надмір інформації також може бути шкідливим і призводити до втрати цілісності, тому інформації потрібно стільки, скільки треба, не більше і не менше.

У процесі обробки даних на ПК широко використовується поняття структури інформації. Структурою визначається будова інформації та передбачається виділення певних її елементів (частин), що називаються одиницями. Одиниці бувають простими й складними. До простих належать такі елементи, які не можна поділити на частини. Складні одиниці — складені, утворені з інших інформаційних одиниць, простих або складних. При ієрархічній (багаторівневій) структурі економічної інформації одиницею нижчого рівня є реквізит, що становить одиницю інформації, яка не підлягає будь-якому членуванню. Реквізити являють собою слова або числа. Реквізити, що характеризують об’єкт управління якісно, називають ознаками, а кількісно — основами. Узяті окремо реквізити — ознаки й основи — не забезпечують всебічної характеристики явищ в економіці. Тому вони об’єднуються, утворюючи таку інформаційну одиницю, як показник. Він може бути простим (складатися з однієї основи й однієї ознаки) або складним (налічувати низку ознак). В управлінні використовуються також одиниці інформації, що складаються з самих реквізитів-ознак. Такі одиниці прийнято називати інформаційними повідомленнями. Інформаційне повідомлення — своєрідний показник, роль основи в якому відіграє провідний рекві­зит — ознака.

Вищий рівень інформаційної одиниці — набір даних, що є сукупністю однорідних показників і реквізитів-ознак на зовнішньому запам’ятовувальному пристрої. Набір даних називається файлом за термінологією ряду систем програмування.

Набір даних (файл) поділяється на частини, які не збігаються з одиницями інформації; сукупність наборів даних, що стосуються однієї ділянки управлінської роботи, часто називають інформаційним потоком.

Будь-які складові інформаційні одиниці (від окремих показників до інформаційної системи в цілому) можна розкладати, зрештою, на реквізити й тим самим підраховувати кількість мінімальних одиниць інформації, що лежать в основі її структурних побудов.

Економічну інформацію класифікують за рядом ознак. Залежно від здійснюваних в управлінні функцій розрізняють планову, облікову та регулюючу інформацію.

Планова в структурі економічної інформації займає 8 – 10%. Вона містить директивні вказівки про розвиток конкретного об`єкта управління та його складових.

Облікова інформація в системі економічної інформації охоплює в середньому 88 – 90%, відображаючи господарські процеси у вигляді натуральних, трудових і вартісних показників. Складовими частинами облікової інформації є бухгалтерські, звітно-статистичні та оперативні дані.

Регулююча інформація займає в середньому майже 2% від загального обсягу економічної інформації. На її основі приймають рішення щодо регулювання параметрів виробництва або планових завдань.

За стадіями утворення економічну інформацію поділяють на  первинну та похідну.

Як первинна, так і похідна економічна інформація може бути змінною (робочою, оперативною) і постійною.

Інформація надходить працівникам апарату управління як із власних структурних підрозділів, так і від інших організацій. За цією ознакою інформацію поділяють на внутрішню і зовнішню.

І, насамкінець, інформацію, що надходить до об`єкта управління, називають вхідною, а інформацію від об`єкта – вихідною.

До економічної інформації пред'являються наступні вимоги: точність, достовірність, оперативність.

*Точність* інформації забезпечує її однозначне сприйняття всіма споживачами.

*Достовірність* визначає допустимий рівень спотворення як вхідної інформації, так і результатної (вихідної), при якому зберігається ефективність функціонування системи.

*Оперативність* відображає актуальність інформації для необхідних розрахунків і прийняття рішень в умовах, що змінилися.

Такі процеси, як збір, зберігання, опрацювання, передача інформації в числовій формі називаються інформаційними і здійснюється за допомогою інформаційних технологій.

Особливістю інформаційних технологій є те, що в них і предметом і продуктом опрацювання є інформація, а знаряддями праці – комп’ютери, засоби обчислювальної техніки і зв'язку.

В системах організаційно-економічного управління інформаційнатехнологія – це такий технологічний процес по опрацюванню інформації, що забезпечує підготовку і прийняття управлінського рішення.

**Формалізація, алгоритмізація та автоматизована обробка економічної інформації**

Підготовка реальних задач до автоматизованого розв’язування з використанням комп’ютера складається з наступних етапів:

І. Постановка задачі та її змістовий аналіз (створення інформаційної моделі задачі).

ІІ. Формалізація задачі (побудова математичної моделі, вибір методу розв’язування)

ІІІ. Складання алгоритму на основі вибраного методу.

IV. Запис програми.

V. Введення, налагодження, тестування програми.

VI. Виконання програми, аналіз результатів.

Формалізації будь-якого процесу, завдання чи задачі передує вивчення структури елементів, з яких складається цей процес, в результаті чого з’являється так званий змістовий опис процесу, тобто основа для створення формалізованої схеми процесу.

Змістовий опис включає в себе:

1. Аналіз умови задачі, а саме, що дано (тобто які величини є вхідними та які їх значення допустимі), що потрібно зробити (тобто, які величини будуть вихідними або результатом і в якому вигляді вони мають бути представлені).

2. Проводиться запис в аналітичній формі всіх співвідношень між вихідними величинами і вхідними, описуються логічні умови та дії щодо реалізації процесу розв’язання.

3. Описується сукупність дій, необхідних для розв’язування задачі, тобто записують алгоритм розв’язування задачі.

Ефективним методом побудови алгоритму є метод покрокової деталізації, при якому завдання розбивається на кілька простих підзадач (модулів) і для кожного модуля створюється свій власний алгоритм. Здебільшого модуль реалізує певний процес обробки інформації і застосовується як окремо, так і для включення модуля в інший алгоритм.

Повторимо.

Застосування *модульності* при створенні алгоритмів дозволяє розбити великі задачі на незалежні блоки (модулі), усуває повторення стандартних дій і значно прискорює процес відлагодження алгоритму в цілому. Найчастіше алгоритм складається з головного модуля, який містить декілька інших модулів, створених раніше. Використовуючи модулі як складові великої конструкції, можна створювати алгоритми будь–якого ступеня складності, і при цьому не втрачати контролю за функціюванням алгоритму всієї задачі. Такий метод називається *структурованим* проектуванням алгоритму „зверху донизу”, є універсальним і може використовуватися як для обчислювальних процесів (так зване системне програмування), так і для процесів реального життя.

Для запису алгоритму рішення задачі застосовуються наступні способи їх подання:

* Словесно- формульний опис
* Блок-схема (схема графічних символів)
* Алгоритмічні мови
* Операторні схеми
* Псевдокод

Для запису алгоритму існує загальна методика:

* Кожен алгоритм повинен мати ім'я, яке розкриває його сенс.
* Необхідно позначити початок і кінець алгоритму.
* Описати вхідні і вихідні дані.
* Вказати команди, які дозволяють виконувати певні дії над виділеними даними

Загальний вигляд алгоритму

* Алгоритм: Назва алгоритму
* Опис даних
* Початок
* Команди
* Кінець

Формульно-словесный спосіб запису алгоритму характеризується тим, що опис здійснюється за допомогою слів і формул. Вміст послідовності етапів виконання алгоритмів записується на природній професійній мові наочної області в довільній формі.

Графічний спосіб опису алгоритму (блок - схема) набув найбільшого поширення.

Для побудови алгоритмів використовують структурний метод, що полягає в структурованому проектуванні алгоритму «зверху-донизу» та метод покрокової деталізації, коли завдання розбивають на кілька простих підзадач (модулів) і для кожної підзадачі (кожного модуля) створюють власний алгоритм, який можна використовувати як окремо, так і включаючи в інший алгоритм, один з яких є головним а інші допоміжні, або підалгоритми. Якщо у процесі обчислення допоміжний алгоритм використовують багаторазово, то його оформляють у вигляді алгоритму-процедури, або алгоритму-функції. Особливість їх використання полягає в тому, що результатом алгоритму-функції є тільки одне значення - значення описаної функції. Комбінуючи базові структури алгоритмів, можна створювати алгоритми будь-якого ступеня складності.

***Для самостійного вивчення*:** Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

***Література***

1. Ковалюк Т. В. Алгоритмізація та програмування: Підручник. — Львів: «Магнолія 2006», 2013. — 400 с., ил.
2. Джейс Либерти Освой самостоятельно С++ за 21 день: 3-е изд. пер. с англ.: Уч. пос. – М.: Издательский дом „Вильямс”, 2001. – 816 с.: ил..
3. Куприянова Л.М. Программирование, алгоритмические языки и вычислительная математика: Учеб. пособие. — М.: Финансы и статистика, 1985. —223 с.

**Контрольні питання**

1. Дати визначення етапів формалізації алгоритмів.
2. Назвіть та охараетермзуйте найпоширеніші методи розробки алгоритмів.
3. В чому полягає аналіз алгоритмів.
4. Цілі структурного програмування.
5. В чому полягає тестування алгоритму.
6. Що розуміють під економічною інформацією і які особливості її оброблення?
7. Чи потрібен змістовний опис для задач оброблення . економічної інформації?