Міністерство освіти і науки України
Національний університет водного господарства та природокористування

Кафедра прикладної математики

04-01-02

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
„Моделювання складних систем”
студентами спеціальності (7.04030101)
“Прикладна математика”
денної форми навчання

Рекомендовано
методичною комісією
зі спеціальності
“Прикладна математика”
Протокол № 11
від 18.06.2014

Рівне 20014

Методичні вказівки до виконання лабораторних робіт з дисципліни “Моделювання складних систем” студентами спеціальності (7.04030101) “Прикладна математика” / Тулашвілі Ю.Й. - Рівне: НУВГП, 2014. – 52 с.

Упорядник:

Ю.Й. Тулашвілі, доктор педагогічних наук, професор кафедри прикладної математики;

Відповідальний за випуск:

Ю.В. Турбал, кандидат фіз-мат. наук, професор, в.о. завідувача кафедри прикладної математики.

ЗМІСТОВИЙ МОДУЛЬ 1 МОДЕЛЮВАННЯ СКЛАДНИХ СИСТЕМ

Тема 2 Декомпозиція. Блочно-ієрархічний підхід до опису складних систем

2.1 Системний підхід до аналізу складних систем

Системний підхід до аналізу складних систем полягає у розгляді об'єкта як системи, яка призначена для виконання чітко встановлених функцій. Це означає, що синтезована у пам'яті комп'ютера математична модель об'єкта є достатньо повною для розробки необхідної проектної документації і містить в собі інформацію про системні аспекти, такі як:

1. Системно-компонентні, які відображають склад елементів і підсистем.
2. Системно-організаційні, що розкривають внутрішню організацію системи, методи взаємовідношень утворюючих її компонентів.
3. Системно-параметричні, які визначають склад і межі допустимих граничних змін параметрів, що характеризують компоненти системи і взаємозв'язок між ними.
4. Системно-функціональні, які визначають характер взаємозв'язків між змінними параметрами.
5. Системно-інтегративні, що показують механізм адаптації об'єкта відповідно до зміни умов його функціонування.

Крім того, ця інформація повинна відображати закономірності розвитку складної системи, як об'єкта проектування та управління.

Виділимо ознаки об'єкта **А** для показу його методами системного аналізу. Об'єкт **А** буде вважатись системним, якщо існують засоби, що дозволяють:

1) поділити об'єкт на фіксовану кількість підсистем першого рівня, відповідно ці підсистеми на підсистеми другого рівня і так до рівня підсистем, які співпадуть із початковими елементами, що не піддаються поділу;
2) визначити системоутворюючі зв'язки, що пов'язують всі елементи та підсистеми існуючих рівнів;
3) визначити параметричні характеристики, які необхідні для виконання аналізу і синтезу підсистем та об'єкта загалом;
4) встановити граничні значення змінних параметрів об'єкта;
5) визначити зв'язки між параметрами та процесом функціонування об'єкта;
6) виявити можливі перетворення, які характеризують самоадаптаційні можливості об'єкта, тобто оцінити процес зміни інградієнтів об'єкта для збереження мінімальної допустимої ефективності його поведінки.

Системно-компонентні і системно-організаційні аспекти об'єкта як системи можна визначити, здійснивши його поділ на фіксовану кількість

підсистем різних рівнів та вихідних елементів з послідовним визначенням системоутворюючих зв'язків між ними. Принцип методу **декомпозиції** проілюстровано на рис. 2.1. Декомпозиція є невід'ємною частиною процесу системного аналізу.

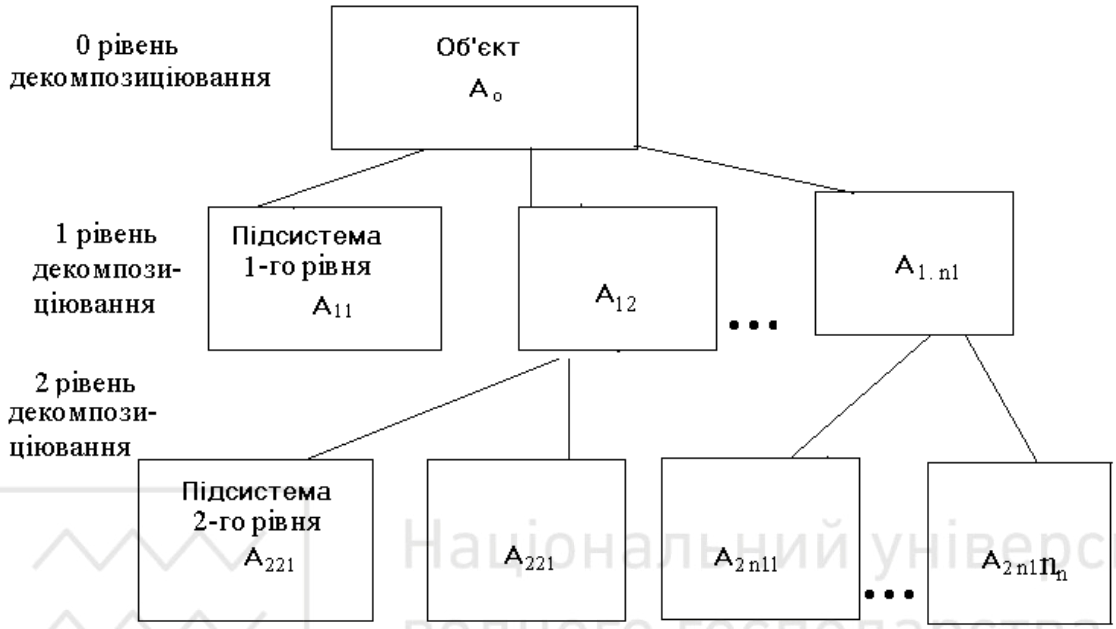


Рис. 2.1. Декомпозиція об'єкта

Внаслідок декомпозиції, яка характеризується різним обсягом охоплення характеристик об'єкта необхідно розрізняти рішення різних рівнів деталізації. Так як що $R_i=\{x_1,x_2,\ldots,x_n\}$ і $R_j=\{y_1,y_2,\ldots,y_m\}$ являють собою i-е та j-е проектні рішення з характеристиками x_i та y_j, то можна казати, що рішення R_i охоплює рішення R_j і навпаки, якщо існує інтервал $x_i \in R_j$. Отримуємо рівність

$$x_i=\{y_i,y_2,\ldots,y_j,\ldots,y_m\}=R_j ,$$

тобто $R_j \in R_i$.

Наприклад, при проектуванні графічної моделі гвинта з головкою із шестигранним заглибленням проектне рішення "головка гвинта" буде охоплювати такі проектні параметри, як "циліндрична поверхня головки" та "шестигранний призматичний отвір". В свою чергу, проектний параметр "шестигранний призматичний отвір" є проектним рішенням, яке містить в собі такі проектні параметри, як "відстань між гранями призматичного отвору" та "висота призматичного отвору".

Всі об'єкти можна відобразити у вигляді абстрактної, субстратно-ієрархічної системи, системи з субстратно- порядковою структурою .

Абстрактна система **S** (рис. 2.2) являє собою множину, яку можна описати такою математичною структурою:

$$S=\{N,R,C,F\},$$

де N – множина – носій системи; R - множина бінарних відношень; C - множини другого порядку бінарних відношень; F - множина відношень порядку.

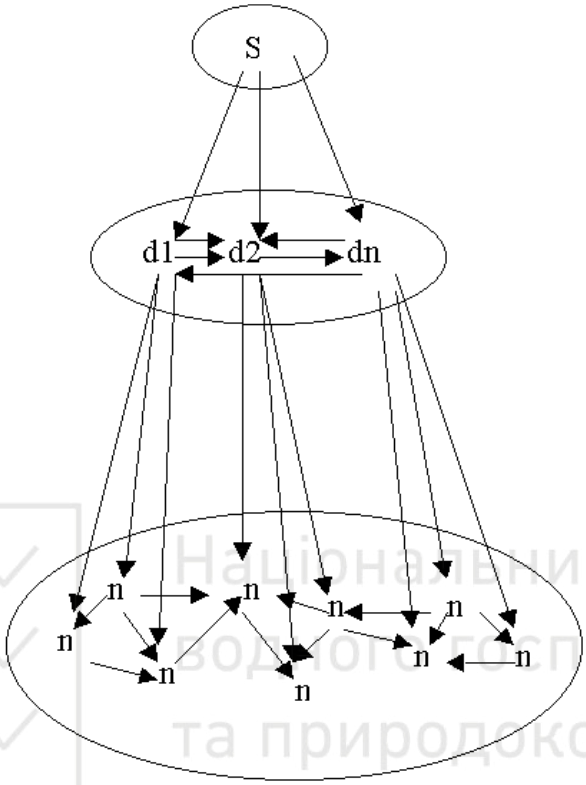


Рис. 2.2. Абстрактна система

Субстрактно-ієрархічна структура системи **S** (рис. 2.3) - це множина:

$$\Sigma S^{def}=\{N,R \mid N\in G\},$$

де G – відношення чітко визначеного часткового порядку.

Між компонентами системи існують різноманітні відношення, що забезпечують цілісність системи в межах виконання її функцій. Характер відношень залежить значною мірою від фізичної природи елементів.

Субстрактно-порядкова структура m-го зрізу системи S (рис. 2.4) являє собою множину:

$$\Sigma S_{пор}{}^{def}=\{N,R_m \mid N\in G,\ R_m\in R\},$$

де R_m – відношення чітко визначеного порядку, що знаходиться у множині бінарних відношень R.

Ефективне застосування програмного забезпечення САПР можливе лише при визначенні комп'ютерних імітаційних моделей як структурних системних одиниць. Створення графічного зображення із застосуванням комп'ютерних

імітаційних технологій пов’язане із розробкою тривимірних моделей. Тривимірні комп’ютерні моделі застосовуються для створення геометричних фігур, форма яких наближена до форми реального об’єкта із відповідними властивостями.

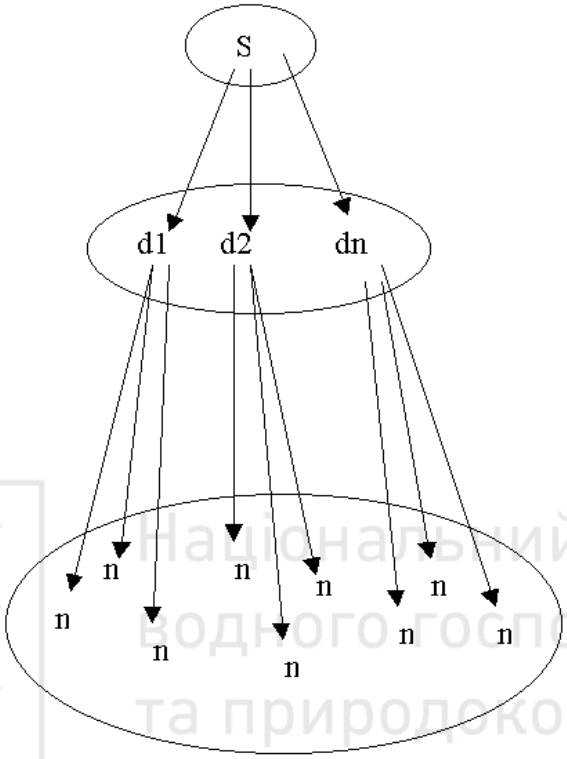


Рис. 2.3.Субстрактно-ієрархічна система

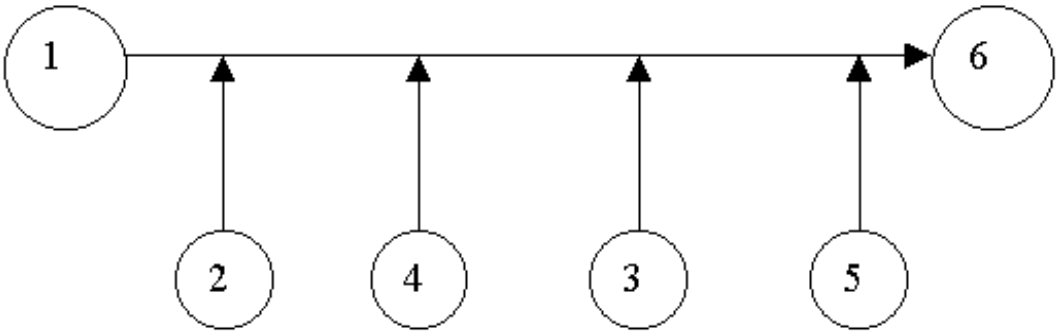


Рис. 2.4. Субстрактно-порядкова система

2.2 Декомпозиція

Блочно-ієрархічний підхід до створення складних систем. Процес розбиття складного об'єкту на порівняно незалежні частини отримав назву **декомпозиції.** При декомпозиції враховують, що зв'язки між окремими частинами мають бути слабкішими, ніж зв'язки елементів усередині частин.

Крім того, аби з отриманих частин можна було зібрати об'єкт, що розробляється, в процесі декомпозиції необхідно визначити всі види зв'язків частин між собою.

При створенні дуже складних об'єктів процес декомпозиції виконується багато разів: кожен блок, у свою чергу, декомпозують на частини, поки не отримають блоки, які порівняно легко розробити. Даний метод розробки отримав назву **покрокової деталізації**.

Важливо і те, що в процесі декомпозиції прагнуть виділити аналогічні блоки, які можна було б розробляти на загальній основі. Таким чином, як вже згадувалося вище, забезпечують збільшення міри повторюваності кодів і, відповідно, зниження вартості розробки.

Результат декомпозиції зазвичай представляють у вигляді схеми ієрархії, на нижньому рівні якої розташовують порівняно прості блоки, а на верхньому - об'єкт, що підлягає розробці.

На кожному ієрархічному рівні опис блоків виконують з певною мірою деталізації, абстагуючись від неістотних деталей. Отже, для кожного рівня використовують свої форми документації і свої моделі, що відображають суть процесів, що виконуються кожним блоком. Так для об'єкту в цілому, як правило, вдається сформулювати лише найзагальніші вимоги, а блоки нижнього рівня мають бути специфіковані так, щоб з них дійсно можна було зібрати працюючий об'єкт.

При дотриманні цього принципу розробник зберігає можливість осмислення проекту і, отже, може приймати найбільш правильні рішення на кожному етапі, що називають **локальною оптимізацією** (на відміну від глобальної оптимізації характеристик об'єктів, яка для дійсно складних об'єктів не завжди можлива).

Метод декомпозиції (сегментування)

На цьому етапі визначається степінь можливого розпаралелення задачі. Іноді декомпозиція поставленої задачі природним чином випливає з самої задачі тому є очевидною, іноді – ні. Чим менший розмір підзадач, отриманих в результаті декомпозиції, чим більша їх кількість, тим гнучкішим виявляється паралельний алгоритм і тим легше забезпечити рівномірне завантаження процесорів обчислювальної системи. Надалі, можливо доведеться укрупнити алгоритм, оскільки слід прийняти до уваги інтенсивність обміну даними та інші фактори. Сегментувати можна як обчислювальний алгоритм, так і дані. Застосовуються різні варіанти декомпозиції.

В методі декомпозиції даних спочатку сегментуються дані, а потім алгоритм їх обробки. Дані розбиваються на сегменти приблизно однакового розміру. З фрагментами даних пов’язуються операції їх обробки, з яких і

формуються підзадачі. Визначаються необхідні правила обміну даними. Перекриття частин, на які розбивається задача, повинне бути мінімальним. Це дозволяє уникнути дублювання обчислень. Схема розбиття може в подальшому уточнюватися. Іноді, якщо це необхідно для зменшення кількості обмінів, допускається збільшення степені перекриття фрагментів задачі.

При декомпозиції даних спочатку аналізуються структури даних найбільшого розміру, або ті, до яких найчастіше відбувається звертання. На різних стадіях розрахунків можуть використовуватися різні структури даних, тому можуть бути необхідними динамічні, тобто такі, що міняються в часі схеми декомпозиції цих структур.

2.3 Методі функціональної декомпозиції при програмуванні

В методі функціональної декомпозиції спочатку сегментується обчислювальний алгоритм, а потім під цю схему підбирається схема декомпозиції даних. Успіх у цьому випадку не завжди гарантовано, оскільки схема може вимагати багатьох додаткових пересилань даних. Цей метод декомпозиції може виявитися корисним у випадку, якщо немає структур даних, які б могли бути розпаралелені очевидним чином. Функціональна декомпозиція відіграє важливу роль і як метод структурування програм. Вона може виявитися корисною при моделюванні складних систем, що складаються з множини простих підсистем, зв'язаних між собою набором інтерфейсів.

Розмір блоків, з яких складається паралельна програма, може бути різним. В залежності від розміру блоків, алгоритм може мати різну "зернистість". Її виміром в найпростішому випадку є кількість операцій у блоці. Виділяють три степені зернистості: дрібнозернистий, середньоблоковий та великоблоковий. Зернистість пов'язана з рівнем паралелізму.

Паралелізм на рівні команд найбільш дрібнозернистий. Його масштаб менше ніж 20 команд на блок. Кількість підзадач, що паралельно виконуються – від однієї до кількох тисяч, при чому середній масштаб паралелізму складає біля п'яти команд на блок.

Наступний рівень - паралелізм на рівні циклів. Переважно, цикл містить не більше 500 команд. Якщо ітерації циклу незалежні, вони можуть виконуватися, наприклад за допомогою конвеєра або на векторному процесорі. Це також дрібнозернистий паралелізм.

Паралелізм на рівні підзадач – середньоблоковий. Розмір блоку – до 2000 команд. Виконання такого паралелізму реалізувати важче, оскільки слід враховувати можливі міжпроцедурні залежності. Вимоги до комунікацій менші, ніж у випадку паралелізму на рівні команд.

Паралелізм на рівні програм (задач) – великоблоковий. Він означає виконання незалежних програм на паралельному комп'ютері. Великоблоковий паралелізм повинен підтримуватися операційною системою.

Обмін даними через спільні/розподілені змінні використовується на рівнях дрібнозернистого і середньоблокового паралелізму, а на великоблоковому – засобами передачі повідомлень.

Досягнути ефективності роботи паралельної програми можна, якщо збалансувати зернистість алгоритму і затрати часу на обмін даними.

Частини програми можуть виконуватися паралельно, лише якщо вони незалежні.

Незалежність за даними полягає в тому, що дані, які обробляються однією частиною програми не модифікуються іншою її частиною.

Незалежність за керуванням полягає в тому, що послідовність виконання частин програми може бути визначена лише під час виконання програми(наявність залежності по виконанню наперед визначає порядок виконання).

Незалежність за ресурсами забезпечується достатньою кількістю комп'ютерних ресурсів(об'єм пам'яті, кількість функціональних вузлів та ін.).

Залежність за виводом виникає, якщо дві підзадачі виконують запис в одну і ту ж змінну. А *залежність за вводом/виводом*, якщо оператори вводу/виводу двох чи декількох під задач звертаються до одного файлу(чи змінної).

Дві підзадачі можуть виконуватися паралельно, якщо вони незалежні за даними, за керуванням і за операціями вводу виводу.

Застосування потоків. У ряді випадків у програмі можна організовувати декілька потоків (ниток), що виконуються одночасно. Наприклад, одна нитка виконання може здійснювати основну роботу, а друга, з меншим пріоритетом, може в той же час готувати або реорганізовувати будь – які файли, малювати зображення, які будуть потрібно надалі, тобто виконувати чорнову роботу. Інший приклад — паралельна робота з декількома зовнішніми джерелами інформації. Особливо великий виграш в продуктивності за рахунок паралельного виконання декількох ниток можна одержати в багатопроцесорних системах, в яких можна організувати виконання кожної нитки окремим процесором.

Паралельно виконувані нитки працюють в адресному просторі одного процесу і можуть мати доступ до глобальних змінних цього процесу.

Одним із способів створення програми з декількома потоками є використовування компоненту типа **TThread.** Цей компонент відсутній в палітрі бібліотеки. **TThread** — це абстрактний клас, що дозволяє створити в

програмі окрему нитку виконання. Для того, щоб ввести **TThread** в свою програму, треба виконати команду File | New | Other. У вікні Депозитарію, що відкрилося, на сторінці New вибрати піктограму Thread Object. З'явиться діалогове вікно, в якому вам буде задано питання про ім'я (Class Name) створюваного класу, TThread**.** Вкажіть будь-яке ім'я, наприклад, Date. Починаючи з Delphi 7, в діалоговому вікні з'явився індикатор Named Thread, який бажано включити, і у полі Named Thread, вказати ім'я потоку, наприклад, MyThread**. Клацніть** ОК, і у ваш проект додасться новий модуль, що має вигляд:

```
unit Unit2;
interface
uses
  Classes // Модуль Windows включається тільки для додатків Windows
{$IFDEF MSWINDOWS}, Windows {$ENDIF};
type
Date = class(TThread)
{ Властивості і методи, що додаються, будуть доступні для функцій
додатку через об'єкт потоку }
  private // Тільки якщо ви включили індикатор Named Thread
procedure SetName;
{ властивості і методи, що додаються тут і у розділі protected будуть
доступні тільки усередині класу }
protected
procedure Execute; override;
end;
implementation
{ Важливо: методи і властивості об'єктів VCL можуть
використовуватися тільки за допомогою запису методу, який
викликається методом Synchronize, наприклад:
  Synchronize(UpdateCaption); де метод UpdateCaption може мати вигляд:
  procedure T. UpdateCaption; begin
  Form1.Caption := 'Updated in a thread'; end; }

{ Тільки для додатків Windows і якщо ви включили індикатор Named Thread
}
{$IFDEF MSWINDOWS}
type
TThreadNamelnfo = record
FType: LongWord;     // повинно бути 0x1000
//покажчик на ім'я в адресному просторі користувача
 FName: PChar;
// ідентифікатор потоку ID (-1  - викликає потік)
    FThreadID: LongWord;
```

```
    // поки не використовується, повинно бути рівно 0
  FFlags: LongWord;
  end;
  {$ENDIF}
{ Date }
{ Тільки для додатків Windows і якщо ви включили індикатор Named Thread
}
procedure T.SetName;
{$IFDEF MSWINDOWS}
var
  ThreadNamelnfo: TThreadNamelnfo;
  {$ENDIF}
  begin
{$IFDEF MSWINDOWS}
ThreadNamelnfo.FType := $1000;
ThreadNamelnfo.FName := 'MyThread'; // вказане ім'я
ThreadNamelnfo.FThreadID := $FFFFFFFF;
ThreadNamelnfo.FFlags := 0;
try
RaiseException ( $406D1388, 0, sizeof(ThreadNamelnfo) div
                  sizeof(LongWord), SThreadNameInfo );
  except
  end;
{$ENDIF}
end;

procedure Date.Execute;
begin
// Тільки якщо ви включили індикатор Named Thread
SetName;
{Place thread code here }
{ Тут розміщується код потоку}
end;
 end.
```

У наведеному вище тексті подано переклад коментарів, які Delphi поміщає в модуль, а також введені коментарі, що пояснюють область видимості нових властивостей і методів, що вводяться вами.

Створений Delphi модуль містить шаблон класу з введеним вами ім'ям класу (в нашому прикладі — Date), що успадковує **TThread.** Ви можете додавати в нього будь-які властивості і методи, враховуючи відзначені в коментарях області видимості. Процедура **Execute,** шаблон якої ви можете бачити в коді, є

основною процедурою нитки. У випадку закінчення завершується і виконання даної нитки додатку.

В процедурі **Execute** можна безпосередньо писати оператори виконання, виклик якихось функцій і таке інше. Проте якщо процедура повинна викликати які небудь методи компонентів VCL або звертатися до властивостей форми, то необхідно дотримуватися обережності, оскільки при цьому не виключені конфлікти між паралельно виконуваними нитками. В цьому випадку в процедурі треба викликати метод **Synchronize**. Метод викликається таким чином:

```
type TThreadMethod = procedure of object;
 procedure Synchronize(Method: TThreadMethod);
```

В цьому визначенні **Method -** процедура, що працює з компонентами VGL. При роботі з компонентами VCL надійніше будувати виконання наступним чином. Ви записуєте процедуру, що виконує необхідні дії з компонентами VCL. Хай ви дали їй ім'я **Work.** Тоді ви включаєте її оголошення в клас нитки, наприклад, в розділ **private,** даєте в розділі **implementation** її опис, а процедура **Execute** в цьому випадку може, наприклад, складатися з єдиного оператора **Synchronize(Work):**


```
Interface
 uses
Classes ...;
type
Data = class(TThread)
 private
{ Private declarations }
   procedure Work;
   protected
   procedure Execute; override;
   end;
. . .
Implementation
. . .
procedure Date.Work;
begin
. . .
end;

procedure T.Execute;
begin
 Synchronize(Work)
 end;
```

Слід врахувати, що в модулі, які створюються Delphi для класу — спадкоємця **TThread,** автоматично в пропозицію **uses** не включаються.

Нормальне завершення виконання потоку відбувається при завершенні процедури **Execute.** Проте можливе і дострокове завершення виконання потоку. Для цього в процедуру **Execute** повинна бути введена булева перевірка властивості **Terminated** (завершено). Нормально ця властивість рівна **false.** Але якщо якась зовнішня нитка викликала метод **Terminate** об'єкту даного потоку, то **Terminated** стає рівним **true.**

Метод **Terminate** забезпечує «м'яке» завершення нитки потоку. Процедура **Execute** сама вирішує, в який момент їй зручно завершити виконання.

Лабораторне заняття № 1

Використання функціональної декомпозиції для розв'язку обчислювальних задач

Мета заняття: навчити студентів методам декомпозицій задач. Набути навиків розв'язування задач з використанням функціональної декомпозиції та організацією декількох потоків (клас Thread) у програмному коді в середовищах Delphi або C++ Builder.

Послідовність виконання:

1. Проаналізувати, наведені нижче, правила обрахунку елементів виразу;
2. Провести декомпозицію задачі, виходячи з можливості паралельного виконання окремих підзадач;
3. Об'єднати отримані проміжні результати і обрахувати кінцевий вираз;
4. Написати і продемонструвати програму обчислення виразу;
5. Визначити паралелізм якого рівня присутній в алгоритмі та зробити висновки щодо залежностей даних, керування, ресурсів, вводу/виводу;
6. Розв'язати завдання запрограмувавши декілька потоків (клас Thread) у програмному коді в середовищах Delphi або C++ Builder
7. Скласти звіт про виконану роботу.

Тема 3 Агрегування. Конфігуратори, оператори. Системні особливості

3.1 Агрегування

Аналіз, як спосіб подолання складності, дозволяє повністю звести складне до простого лише у випадку складності через непоінформованість (шляхом залучення додаткових експертів); у випадку складності, що виникає через нерозуміння, аналіз не дозволяє звести складне до простого, але локалізує її.

Агрегування — це операція об'єднання декількох, елементів в єдине ціле, протилежна до декомпозиції. Об'єднані елементи, що взаємодіють між: собою, набувають не лише зовнішньої, але й внутрішньої цілісності, єдності. Зовнішня цілісність відображається моделлю «чорної скрині», а внутрішня — пов 'язана зі структурою системи, і виявляється в тому, що властивості системи є більшими, ніж сума властивостей об'єднаних елементів.

Види агрегатів СС

Техніка агрегування грунтується на використанні певних моделей системи, а саме: модель складу, яка визначає, що повинно ввійти до складу системи та модель структури, яка відображає зв'язки елементів між собою.

У процесі декомпозиції вирішення цієї проблеми досягалося шляхом компромісу — за допомогою поняття суттєвості, що супроводжувалося ризиком недостатньої повноти чи зайвої деталізації, то в процесі агрегації проблема ускладнюється, тому що ризик неповноти є майже неприпустимим. Виходячи з цього виникло поняття конфігуратора.

Конфігуратор. Будь-яке дійсно складне явище вимагає сумісного (агрегованого) описання в термінах декількох якісно відмінних мов. Конфігуратором будемо вважати агрегат, що складається з якісно різних мов описання системи, причому кількість цих мов є мінімально необхідною для досягнення мети.

Головним в конфігураторі є не те, що аналіз об'єкта повинен проводитися кожною мовою конфігуратора окремо, а те, що синтез можливий лише при наявності всіх описів.

Конфігуратор є змістовною моделлю найвищого рівня. Перерахувавши мови, якими ми будемо описувати систему, ми тим самим визначаємо, синтезуємо тип системи, фіксуємо наше розуміння природи системи. Як і будь-яка модель, конфігуратор має цільовий характер і при зміні мети може втратити властивості конфігуратора.

Досить часто виникають ситуації, в яких сукупність даних, якими необхідно оперувати, дуже чисельна, внаслідок чого з ними складно і незручно працювати. Саме це і приводить до необхідності агрегування — в цьому випадку на перше місце висувається така особливість агрегування, як

зменшення розмірності, і агрегат об'єднує частини в дещо ціле, єдине та окреме.

В цьому випадку ми розглядаємо **агрегати-оператори**. Найпростіший спосіб агрегування полягає у встановленні відношення еквівалентності між елементами, що підлягають агрегації, тобто утворення класів. Класифікація є дуже важливим, багатобічним, багатофункціональним явищем, і з практичної точки зору важливими проблемами є як визначення класів, так і визначення, до якого класу належить той чи інший конкретний елемент.

Якщо класифікація має природний характер, то агрегування побічних ознак може розглядатися як виявлення загальних закономірностей в таблицях експериментальних даних.

У випадку, коли ознаки, що агрегуються, вимірюються в числових шкалах, може виявитися можливим задати відношення на множині ознак у вигляді числової функції багатьох змінних, яка й буде агрегатом.

Агрегат-оператор дозволяє зменшити розмірність інформації, але при його застосуванні слід вважати на можливі наступні негативні особливості:

– втрата корисної інформації, оскільки агрегування є необоротним перетворенням (найпростіший приклад — за сумою неможливо повернутися до значень її складових);

– агрегування — це вибір певної визначеної моделі системи, з чим пов'язані непрості проблеми оцінки адекватності;

– для деяких агрегатів властива внутрішня суперечність (приклад — парадокс голосування Ероу).

Агрегати-структури. Як і будь-який інший вид агрегату, структура є моделлю системи і визначається об'єктом, метою та засобами моделювання.

У процесі синтезу ми створюємо структуру майбутньої системи, що проектується.

Тому при проектуванні системи важливо задати структури в її суттєвих відношеннях.

3.2 Агрегування при написанні програмного коду

В основу ООП закладені абсолютно інші принципи і інша стратегія написання програм, що часто стає каменем спотикання для багатьох розробників, що звикли до традиційного програмування. Тепер програми є алгоритмами, що описують взаємодію груп взаємозв'язаних об'єктів.

У С++ об'єкти створюються шляхом опису КЛАСУ як нового типу даних. Клас містить ряд констант і змінних (даних), а також операцій (функцій-членів, або методів), що виконуються над ними.

Щоб виконати яку-небудь дію над даними об'єкту, йому необхідно, виражаючись в нових термінах, послати повідомлення, тобто викликати один з його методів.

До даних, що зберігаються в об'єкті, не можна дістати доступ інакше, як шляхом виклику того або іншого методу. Таким чином, програмний код і оперовані дані об'єднуються в єдиній "віртуальній" структурі.

ООП дає програмістам три важливі переваги.

Перше полягає в спрощенні програмного коду і поліпшенні його структуризації. Програми сталі простіше для читання і розуміння.

Код опису класів, як правило, відокремлений від коду основної частини програми, завдяки цьому над ними можна працювати окремо.

Друга перевага полягає в тому, що модернізація програм (додавання і видалення програмних блоків) стає незрівняно простішим завданням. Зазвичай вона зводиться до додавання нового класу який наслідує усі властивості одного з наявних класів і містить необхідні додаткові методи.

Третя перевага полягає в тому, що одні і ті ж класи можна багато разів використовувати в різних програмах. Вдало створений клас можна зберегти окремо в бібліотечному файлі, і його додавання в програму, як правило, не вимагає внесення серйозних змін до її тексту.

Основна термінологія

Оскільки ідея ООП є досить глобальною і не пов'язаною з конкретною мовою програмування, потрібно було визначити терміни, загальні для усіх мов програмування, що підтримують цю концепцію. Тобто ті терміни, з якими ми зараз познайомимося, є загальними як для Pascal, так і для C++ і інших мов.

Втім, разом із загальною термінологією існують і терміни, специфічні для окремих мов.

Клас мови C++ є розширеним варіантом структури і слугує для створення об'єктів. Він містить функції-члени (методи), пов'язані деякими загальними атрибутами.

Об'єкт - це екземпляр класу, що є доступний для виконання над ним необхідних дій.

Інкапсуляція. Під інкапсуляцією розуміється зберігання в одній структурі як даних (констант і змінних), так і функцій їх обробки (методів).

Доступ до окремих частин класу регулюється за допомогою спеціальних ключових слів **public** (відкрита частина), **private** (закрита частина) і **protected** (захищена частина).

Методи, розташовані у відкритій частині, формують інтерфейс класу і можуть вільно викликатися усіма користувачами класу.

Доступ до закритої секції класу можливий тільки з його власних методів, а до захищеної - також з методів класів-нащадків.

Ієрархія класів. У С++ клас виступає шаблоном, на основі якого створюються об'єкти. Від будь-якого класу можна породити один або декілька підкласів, внаслідок чого сформується ієрархія класів. Батьківські класи зазвичай містять методи загальнішого характеру, тоді як рішення специфічних завдань доручається похідним класам.

Спадкоємство.Під спадкоємством розуміють передачу даних і методів від батьківських класів до класів-нащадків (похідних).

Якщо клас наслідує свої атрибути від одного батьківського класу, то таке спадкоємство називається поодиноким. Якщо ж атрибути наслідують від декількох класів, то говориться про множинне спадкоємство. Спадкоємство є найважливішою концепцією програмування, оскільки дозволяє багаторазово використовувати одні; і ті ж класи, не переписуючи їх, а лише підлаштовуючи для вирішення конкретних завдань і розширюючи їх можливості.

Поліморфізм. Інша важлива концепція ООП , пов'язане з ієрархією класів, полягає в можливості послати однакове повідомлення відразу декільком класам в ієрархії, надавши їм право вибрати, кому з них належить його обробити. Це називається поліморфізмом.

Віртуальні функції. Методи, що містяться в різних класах однієї ієрархії, але що мають загальне ім'я і оголошені з ключовим словом virtual, називаються віртуальними.

Завдяки поліморфізму можна робити в програмі запит до об'єкту, навіть якщо тип його не відомий заздалегідь. У С++ ця можливість реалізується за рахунок підсистеми пізнього зв'язування, під яким розуміється динамічне визначення адрес функцій під час виконання програми в протилежність традиційному статичному (ранньому) замінюються їх адресами.

При виклику віртуальних функцій використовується спеціальна таблиця адрес функцій, звана віртуальною таблицею. Вона ініціалізувалася в ході виконання програми у момент створення об'єкту конструктором класу. Роль конструктора полягає в тому, щоб зв'язати віртуальну функцію з правильною таблицею адрес. Під час компіляції адреса віртуальної функції не відома, але відомий елемент віртуальної таблиці, де ця адреса буде записана під час виконання програми.

Синтаксис опису класів

Синтаксис опису класу подібний до синтаксису опису структури. Воно починається з ключового слова **class**, за яким йде ім'я класу, що стає ім'ям нового типу даних. У простому випадку опис класу можна представити так:

class ім'я {

```
тип1 переменная1;
тип2 переменная2;
тип3 переменная3;

public:
 метод1;
метод2;
 метод3;
};
```

За умовчанням усі члени класу вважаються закритими і доступ до них можуть отримати тільки функції-члени цього ж класу.

Це саме те, що в C++ відрізняє класи від структур: усі члени структур за умовчанням є відкритими.

Якщо необхідно змінити тип доступу до членів класу, перед ними слід вказати один із специфікаторів доступу : public, protected або private.

Приклад простого класу

Розглянемо наступний приклад програми :

```
// MyClass.cpp
#include "stdafx.h"
#include <iostream>
#include <locale>
#include <math.h>
const double DEG_TO_RAD = 0.0174532925;
class degree {
double data_value;
public:
void set_value (double angle);
double get_sine(void);
double get_cosine(void);
double get_tangent(void);
double get_secant(void);
double get_cosecant(void);
double get_cotangent(void); } deg;
void degree::set_value (double angle) {  data_value = angle;  }
double degree::get_sine(void) { return(sin (DEG_TO_RAD * data_value)); }
double degree::get_cosine(void) { return(cos (DEG_TO_RAD * data_value));}
double degree::get_tangent(void){ return(tan(DEG_TO_RAD * data_value)); }
double degree::get_secant(void) { return(1.0 / sin(DEG_TO_RAD * data_value)); }
double degree::get_cosecant(void) {return (1.0/ cos(DEG_TO_RAD * data_value));}
```

```
double degree::get_cotangent(void) { return(1.0/ tan(DEG_TO_RAD * data_value));}
int _tmain(int argc, _TCHAR* argv[])
        {
        setlocale(0,"");
          // встановлюємо значення кута рівним 25.0градусов
        deg.set_value (25.0);
        std::cout << "Синус кута рівний " << deg.get_sine() << std::endl;
        std::cout<< "Косинус кута рівний " << deg.get_cosine() << std::endl;
        std::cout << "Тангенс кута рівний " << deg.get_tangent() << std::endl;
        std::cout << "Секанс кута рівний " << deg.get_secant() << std::endl;
        std::cout << "Косеканс кута рівний " << deg.get_cosecant() << std::endl;
        std::cout << "Котангенс кута рівний " << deg.get_cotangent() << std::endl;
          system("pause"); // Команда задержки экрана
         return 0;
        }
```

Тут створюється новий тип даних degree. Закрита змінна-член data_value доступна тільки функціям-членам класу. Одночасно створюється і об'єкт цього класу - змінна deg.

Це, по суті, та ж сама структура, яку ми розглядали в попередньому прикладі, лише ключове слово **class** перетворює структуру на справжній клас.

Слід звернути увагу на синтаксис заголовка функції, наприклад:

void degree::set_value(double angle)

Ім'я функції складається з імені структури, за яким розташований оператор дві двокрапки "**::**".

При виклику функції використовується оператор крапка "**.**", як і при доступі до змінних-членів структури.

Якби структура була представлена покажчиком, доступ до функцій необхідно було б здійснювати за допомогою оператора стрілочка "**->**".

Програма виведе на екран наступну інформацію:

Синус кута дорівнює 0.422618

Косинус кута дорівнює 0.906308

Тангенс кута дорівнює 0.466308

Секанс кута дорівнює 2.3662

Косеканс кута дорівнює 1.10338

Котангенс кута дорівнює 2.14451

Конструктори. Одне з основних завдань об'єктно-орієнтованого програмування полягає у тому, щоб об'єкти описаного раз і назавжди класу працювали «правильно» — тобто так, як це визначає модель. Кожний об'єкт

перед тим як почати роботу, потрібно створити, тобто перевести в якийсь початковий стан. Отже, треба якимось чином описати можливі механізми створення об'єктів даного класу. Для цього в мові С++ існують ***конструктори***. Це особливі методи класу, які й повинні перевести об'єкт у той самий початковий стан. Конструктор описується як метод, ім'я якого збігається з іменем класу, а тип поверненого значення опущений.

Приклад:

```
class Point
{
  public:
    Point(int x0, int y0);
  private:
    int x, y;
};
Point::Point(int x0, int y0)
{
  x=x0;
  y=y0;
}
```

Тепер для створення об'єкта класу Point потрібно після імені змінної вказати параметри, як для виклику функції:

```
Point A(1, 1), B(2, 0);
```

Типи конструкторів. Існують деякі типи конструкторів, які, крім безпосереднього використання, автоматично викликаються у деяких особливих ситуаціях.

Конструктор за замовчуванням. Це конструктор, що викликається без параметрів:

```
Point();
Point(int a=5);
```

Його використовують для створення масиву об'єктів, оскільки не зрозуміло, які конструктори і з якими параметрами треба викликати для кожного елемента масиву. Наприклад:

```
Point A[10];
Point* B=new Point[10];
```

Конструктор за замовчуванням викликається також тоді, якщо не вказано параметри для ініціалізації об'єкта, як у цьому випадку:

```
Point p;
```

Конструктор копіювання. Цей конструктор викликається тоді, коли потрібно створити копію об'єкта. Аргументом цього конструктора має бути посилання на об'єкт цього самого класу:

Point(Point& p);

Важливим випадком, коли викликається конструктор копіювання, є передавання об'єкта у функцію як параметра за значенням. Тоді створюється новий об'єкт і для нього автоматично викликається конструктор копіювання. Створення конструкторів копіювання потрібне у випадку, якщо об'єкт потребує якихось спеціальних операцій при копіюванні, оскільки під час стандартного копіювання вміст одного об'єкта просто побайтно переноситься в інший.

Приклад:

```
class String
{
public:
  String();                    // конструктор за замовчуванням
  String(const String& s); // конструктор копіювання
  String(const char* s);     // конструктор з параметром
                                    // const char*, який являє собою
                                    // стандартний рядок  s
  ~String();            // деструктор
private:
  char* array;              // масив символів
  int size;                  // розмір масиву
};
```

Приклад виклику конструкторів:

```
int main()
{ String a, b;  // конструктор за замовчуванням
  String c(a);  // конструктор копіювання
  print(a);     // конструктор копіювання, оскільки
                          // аргумент передається у функцію за значенням
  String d("One");   // конструктор з параметром
//...
}
```

Деструктори. Конструктори ініціалізують об'єкт, тобто вони створюють середовище, у якому "працюють" функції-члени. Іноді створення такого середовища зумовлює "захоплення" якихось ресурсів: пам'яті, файлу, процесорного часу, які повинні бути "звільнені" після їх використання. Тобто класам потрібна функція, яка б знищувала об'єкт аналогічно тому, як його створює конструктор. Такі функції називають ***деструкторами.***

Приклад:

```
class Name
 {
    const char* s;
    // ...
 };

class Table
```

```
{
  Name* p;
  size_t sz;
 public:
  Table(size_t s=15) {p=new Name[sz=s];}
                                                    //конструктор
  ~Table() {delete[] p;}                  //деструктор
// ...
 };
```

Лабораторна робота № 2

Агрегування засобами програмування. Проектування класів та їх методів для створення відношення агрегації

Мета заняття: засвоїти поняття конструктора, деструктора та функцій-членів класів.

Послідовність виконання:

Згідно з номером варіанта спроектувати класи, реалізувавши конструктори та відповідні методи. В головній функції проілюструвати їх використання.

Варіанти завдань

1. ***Динамічний список (2 класи: елемент списку, список перебувають у відношенні агрегації)***

Конструктори: за замовчуванням, з параметрами та копіювання.

Деструктор.

Функції (дві-три на вибір):

вставлення елемента з голови (хвоста) у заданому місці;

видалення елемента з голови (хвоста) із заданого місця;

виведення списку на екран;

пошук елемента списку;

інформація про кількість елементів списку;

очищення списку;

одержання голови (хвоста) списку;

одержання наступного (попереднього) елемента.

2. ***Множина (2 класи: елемент множини, множина перебувають у відношенні агрегації)***

Конструктори: за замовчуванням, з параметрами та копіювання.

Деструктор.

Функції (дві-три на вибір):

додавання елемента до множини;

видалення елемента із множини;

виведення всіх елементів множини на екран;

перевірка входження елемента до множини;

очищення множини;

перебирання всіх елементів множини.

3. *Текст (2 класи: рядок, текст перебувають у відношенні агрегації)*

Конструктори: за замовчуванням, з параметрами та копіювання.

Деструктор.

Функції (дві-три на вибір):

вставляння (видалення) символу у даному рядку, у даному місці;

очищення заданого рядка;

видалення заданого рядка;

вставляння рядка у задане місце;

пошук рядка у тексті;

очищення тексту.

Тема 4 Аксіоматичний підхід дослідження складних систем. Метод чорної скриньки.

4.1 Аксіоматичний підхід дослідження систем.

Аксіоматичний підхід є одним з найбільш поширених підходів при формальному дослідженні систем. Його особливістю є те, що модель будується на певних базових припущеннях, які не вимагають теоретичного обґрунтування – аксіомах.

Окрім базових припущень про внутрішні системні механізми, що вивчаються, при аксіоматичному підході важливим є припущення про достатність математичної моделі для досягнення мети моделювання.

Основними вимогами до базових припущень є:

• несуперечність системи аксіом;

• абстрактність, яка включає терміни та символи формальної мови, вислови на них побудовані, які формують математичні вирази для опису характерних властивостей системи чи правила виведення нових виразів.

Базові припущення формуються на основі змістовного (вербального) описання функціонування системи. Процес побудови аксіоматичної моделі

вимагає інтерпретації та переведення змістовного описання на мову строгих математичних відношень та термінів. При цьому усуваються багатозначність трактування, неповнота, неясність та неконкретність властиві змістовному описанню.

Послідовність досліджень при аксіоматичному підході.

1. Відображення уявлень дослідників про систему за допомогою змістовного опису системи.

2. Формалізація змістовного опису та побудова системи аксіом – як уявлень про майбутню модель системи.

3. Отримання моделі системи на основі аксіом шляхом гомоморфного відображення реальних властивостей системи за допомогою формалізованого виведення.

4. Інтерпретація моделі на основі пояснення теоретичних результатів – як відображення результатів діяльності реальної системи.

5. Перевірка достовірності, точності, повноти моделі та встановлення меж змістовної відповідності.

6. Побудова теорії за результатами інтерпретації та визначення меж її застосування. Пояснення за допомогою теорії відомих фактів поведінки системи.

7. Застосування теорії з метою виявлення нових властивостей системи.

8. Експериментальне підтвердження отриманих результатів застосування теорії.

Аксіоматичний підхід добре зарекомендував себе при побудові детермінованих моделей та при розв’язуванні проблем, що можуть бути строго формалізовані. Складні ж проблеми, є слабко структурованими і не можуть бути повністю формалізовані. Як правило аксіоматичний підхід можна з успіхом застосовувати для дослідження окремих підсистем та елементів у детермінованому середовищі.

Метод “чорної скриньки”. Невизначеність при побудові моделей “вхід вихід”.

Кібернетичне трактування неможливості повної ідентифікації усіх властивостей системи, її структури втілилося в ідеї “чорної скриньки”.

Моделі “чорної скриньки” дозволяють відобразити ті входи та виходи системи, що необхідні для вивчення однієї з сторін її функціонування, тому називаються моделями “вхід-вихід”. При побудові такої моделі встановлюється відношення між цими входами та виходами. Модель “вхід-вихід” відображає основні властивості системи, такі як цілісність та відносну ізольованість через наявність зв’язок із зовнішнім середовищем.

При побудові моделі “вхід-вихід” основною проблемою є визначення тих входів та виходів, які необхідно включати до складу моделі, оскільки при вивченні системи модель постійно модифікується. Реальна система взаємодіє із середовищем через нескінченну кількість способів, тобто через нескінченну кількість входів та виходів. Критерієм відбору цих входів та виходів є цільове призначення системи, суттєвість того чи іншого зв’язку системи із середовищем. У моделі ми вимушені відображати скінченну кількість взаємодій і тим самим існує висока ймовірність не включення саме тих входів та виходів, які найбільш суттєво визначають властивості системи.

При побудові моделей “вхід-вихід” невраховані та невідомі зв’язки із зовнішнім середовищем представляють за допомогою спрощених моделей невизначеності. Сучасні підходи до побудови моделей “вхід-вихід” побудовані на відображені однієї з таких форм невизначеності: стохастичної та теоретико-множинної, або їх комбінації.

У цих випадках модель система розглядається у вигляді “чорної скриньки”, зображеної на рис.

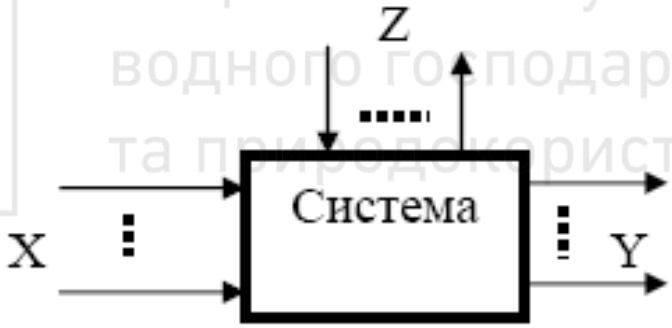


Рисунок 4.1 – Метод “чорної скриньки”

У випадку прийняття гіпотези про випадкову (імовірнісну) природу дії неврахованих та невідомих зв’язків із зовнішнім середовищем використовують стохастичну форму невизначеності. Суть її у кількісному виразі зводиться до того, що дія середовища на систему і системи на середовище відображається у випадкових змінах характеристик контрольованих і врахованих зв’язків. Для дослідження випадкових змін цих характеристик використовують методи статистичного оцінювання.

Залежно від способу опису форм невизначеності при побудові моделей “вхід-вихід” використовують різні підходи до моделювання: детермінований, стохастичний та теоретико-множинний чи інтервальний. Кожен із вказаних підходів визначатиме відповідну методологію побудови моделей “вхід-вихід”.

Одним з наукових методів оцінювання стану функціональності складної системи та прийняття рішення є **експертний метод**.

В процесі оцінювання ефективності складних систем дуже часто чинники, які слід враховувати є настільки нові й складні, що достатня інформація оцінювання про них відсутня, а ймовірність того чи іншого результату не можна обчислити статистичними методами.

Стикаючись з потребою оцінити стан функціональності складної системи та прийняти рішення в поточній ситуації, можна спробувати дістати додаткову інформацію, ще раз проаналізувавши проблему. Для отримання більш об'єктивної оцінки у такому випадку застосовується метод ***експертних оцінок***.

Кваліфіковані експерти здатні вивести проблему пошуку рішення з розряду невизначеності до розряду стохастики, надавши проблемі вибору наукової та практичної обґрунтованості.

На праці є доцільним, щоб експерти дали свої оцінки ймовірностей виникнення певних подій, на підґрунті яких можна було б відшукати середні значення експертних оцінок і на їх підставі побудувати криву розподілу суб'єктивних ймовірностей. Наприклад, за допомогою експертних оцінок потрібно знайти показники найбільш імовірних, допустимих, критичних і катастрофічних збитків, маючи на увазі як їх обсяги, так і відповідні суб'єктивні ймовірності.

Експертний метод можна реалізувати, опрацювавши судження досвідчених фахівців. Бажано, щоб експерти супроводжували свої оцінки даними щодо ймовірності виникнення різних значень (обсягів чи відсотків).

Експерт відіграє роль певного приладу, який або робить пріоритетний вибір, або встановлює логічний зв'язок, що об'єктивно існує між причиною і наслідком. Таким чином, суб'єктивна ймовірність дає змогу встановити зв'язок між невизначеністю та випадковістю.

Найприйнятнішим варіантом для розв'язування практичних проблем є, очевидно, комбінація статистичного та експертного способів здобуття інформації.

Загальна схема експертизи, підготовка до її проведення. Експертиза як метод здобуття інформації завжди використовувалась у процесі оцінювання та розробці рішень.

Сутність методу експертних оцінок полягає в раціональній організації експертного аналізу проблеми з кількісним оцінюванням суджень і обробкою їх результатів. Узагальнену думку експертів вважають рішенням проблеми.

Усе розмаїття розв'язуваних задач зводиться до трьох типів:

1) формування об'єктів;
2) оцінювання характеристик;

3) формування об’єктів і оцінювання характеристик.

Формування об’єктів передбачає визначення можливих подій і явищ, побудову гіпотез, формування цілей, обмежень, варіантів рішень, визначення ознак і показників для опису властивостей об’єктів та їх взаємозв’язків тощо. Вирішуючи задачу ***оцінювання характеристик***, експерти визначають імовірність подій, важливість цілей, значення ознак і показників, переваги рішень. Задача ***формування об’єктів та оцінювання характеристик*** охоплює комплексне розв’язування перших двох типів задач. Отже, експерт виконує роль генератора об’єктів (ідей, подій, рішень) та вимірювача їх характеристик.

Щоб застосувати метод експертних оцінок у процесі прийняття рішень, необхідно розглянути питання добору експертів, провести їх опитування й обробити одержані результати. Залежно від масштабу проблеми експертизу організовує суб’єкт управління або призначена ним група. Кількісний та якісний склад експертів добирають на підставі аналізу широти проблеми, вірогідності оцінок, характеристик експертів і затрат ресурсів.

Різноманітність розв’язування проблеми зумовлює потребу залучати до експертизи фахівців різного профілю. Мінімальна кількість експертів визначається кількістю різноманітних аспектів, напрямів, які потрібно враховувати, вирішуючи проблему.

Характеристики групи експертів визначаються на базі індивідуальних характеристик експертів: компетентності, креативності, конформізму, ставлення до експертизи, конструктивності мислення, колективізму, самокритичності.

Компетентність — це ступінь кваліфікації експерта в певній галузі знань, який можна визначити, проаналізувавши діяльність фахівця, рівень і широту ознайомленості з досягненнями світової науки та техніки, розуміння перспективи розвитку.

Креативність — це здатність розв’язувати творчі задачі. Досі, крім якісних суджень, що ґрунтуються на вивченні діяльності експертів, немає жодних точних способів оцінювання цієї характеристики.

Конформізм — це врахування впливу авторитетів. Особливо виразно конформізм може виявитись під час проведення експертизи у вигляді відкритих дискусій. Думка авторитетів пригнічує здатність осіб, що мають високий ступінь конформізму, виявити власне ставлення до проблеми.

Ставлення до експертизи — дуже важлива характеристика якості експерта при розв’язуванні даної проблеми. Негативне або пасивне ставлення фахівця до тієї чи іншої проблеми, зумовлене зайнятістю та іншими чинниками, істотно позначається на виконанні експертами своїх функцій. Тому участь в експертизі має розглядатись як планова робота.

Конструктивність мислення — це прагматичний аспект мислення. Експерт має пропонувати розв'язки з практичними властивостями. Урахування реальних можливостей розв'язання проблеми — дуже важливий чинник у проведенні експертного оцінювання.

Колективізм — це чинник, який має враховуватись під час проведення відкритих дискусій. Етика, поведінка людини в колективі істотно впливає на створення позитивного психологічного клімату і тим самим на успішність розв'язання проблеми.

Самокритичність експерта виявляється у самооцінюванні ступеня своєї компетентності, а також у прийнятті рішення з досліджуваної проблеми.

Перелічені характеристики експерта досить повно описують важливі якості, які впливають на результати експертизи. Частина їх оцінюється позитивно, частина — негативно. Виникає проблема узгодження характеристик та вибору експертів з урахуванням суперечливостей щодо їх якостей. Часто вірогідність (правильність) оцінок експерта кількісно можна оцінити за формулою

$$D_i = \frac{N_i}{N}, \quad i = 1, \quad ..., \quad m,$$

де N_i — кількість випадків, коли *i*-й експерт дав оцінку, яка підтвердилась практикою;

N — загальна кількість випадків участі *i*-го експерта у розв'язуванні проблеми.

Частка внеску кожного експерта у вірогідність оцінок усієї групи обчислюється як:

$$D_i' = \frac{D_i}{\sum_{i=1}^{m} D_i}, \quad i = 1, ..., m,$$

де *m* — кількість експертів у групі.

Опитування — основним етапом спільної роботи груп управління та експертів на цьому етапі є такі процедури:

- вирішення організаційно-методичних питань;
- постановка задачі та формування запитань до експертів;
- інформаційне забезпечення роботи експертів.

Вибір того чи іншого виду опитувань визначається цілями експертизи, сутністю вирішуваної проблеми, повнотою та вірогідністю початкової інформації, терміном і затратами на проведення опитування. Розглянемо зміст і технологію проведення таких видів опитування, як анкетування та інтер-в'ювання.

Анкетування — це опитування експертів у письмовій формі за допомогою анкети, в якій містяться запитання, котрі можна класифікувати за змістом і типом. За змістом вони поділяються на три групи:

- об'єктивні дані про експерта (вік, освіта, посада, фах, стаж роботи);
- основні запитання по суті розв'язуваної проблеми;
- додаткові запитання, що дають змогу знайти джерела потрібної інформації, аргументацію відповідей, самооцінку компетентності експерта.

Якщо анкетування проводиться кількома турами, доцільно за великої складності та невизначеності проблеми спочатку використати відкриті типи запитань, а на наступних турах — альтернативні відповіді та закриті типи.

Інтерв'ювання — це усне опитування, що проводиться у формі бесіди-інтерв'ю. Для підготовки бесіди інтерв'юєр розробляє запитання до експерта. Характерна особливість цих запитань — на них експерту потрібно відповісти швидко, оскільки практично немає часу на їх обмірковування. Тематику інтерв'ю експерту можна повідомити заздалегідь, але конкретні запитання слід сформулювати безпосередньо в процесі бесіди. Доцільно в зв'язку з цим готувати послідовність запитань, починаючи від простого і поступово поглиблюючи та ускладнюючи, але водночас конкретизуючи запитання.

Перевагою інтерв'ю є живий контакт інтерв'юєра з експертом, що дає змогу оперативно дістати необхідну інформацію з допомогою прямих та уточнюючих запитань — залежно від відповідей експерта.

Недоліком інтерв'ю є небезпека великого впливу інтерв'юєра на відповіді експерта, відсутність часу для глибокого обмірковування відповіді.

Метод Дельфі є багатотуровою процедурою анкетування. Назва Дельфі (Дельфійського методу) запозичена з історії Дельфійського оракула. У першому турі експертам пропонуються запитання, на які вони дають відповіді з метою виокремлення середнього, медіани і граничних значень оцінок. Експертам повідомляють результати обробки першого туру опитування, уточнюючи розміщення оцінок кожного експерта. Якщо оцінка експерта дуже відхиляється від середнього значення, то його просять аргументувати свою думку або змінити оцінку.

У другому турі експерти аргументують або змінюють свою оцінку з поясненням причин коригування. Результати обробляють і повідомляють експертам. Якщо після першого туру коригувались оцінки, то результати обробки другого туру містять нові середні та граничні значення оцінок експертів.

Наступні тури проводять за аналогічною процедурою. Звичайно після третього чи четвертого туру оцінки експертів стабілізуються, що і є критерієм припинення подальшого опитування.

Дослідження ефективності методу Дельфі показали, що з проведенням турів опитування розкид думок експертів зменшується і групова думка у вигляді медіани індивідуальних оцінок стає точнішою.

Щоб підвищити ефективність проведення експертизи за методом Дельфі, необхідно автоматизувати процес фіксування, обробки та повідомлення експертам інформації. Це досягається завдяки використанню ЕОМ.

Інтелектуальний штурм — це групове обговорення, що має на меті здобуття нових ідей, варіантів вирішення проблеми. Інтелектуальний штурм часто називають також інтелектуальною атакою, методом генерування ідей. Характерною особливістю цього виду експертизи є активний творчий пошук принципово нових рішень, коли відомі шляхи та способи рішень непридатні. Щоб активність експертів не згасала, категорично забороняється критика їхніх висловлювань.

Експерти складають групу до 20-25 осіб, в яку залучаються фахівці з розв'язуваної проблеми та люди з глибокою ерудицією і багатою фантазією, причому не обов'язково добре обізнані з досліджуваного питання. Для проведення сеансу призначається головуючий, основне завдання якого — керувати перебігом обговорення поставленої проблеми. Він на початку сеансу пояснює зміст і актуальність проблеми, правила її обговорення і пропонує для розгляду одну-дві ідеї.

Сеанс триває приблизно 40-45 хв. без перерви. Для кожного окремого виступу надається 2-3 хв. За цей час експерти мають спробувати висунути якомога більше нових, на перший погляд, фантастичних ідей або розвинути висловлені ідеї, доповнивши та поглибивши їх. Важливою вимогою до виступів є конструктивний характер ідей і пропозицій. У процесі генерування ідей та їх обговорення пряма критика заборонена, проте неявно присутня весь час і виражається ступенем підтримки та розвитку висловлювань.

Виступи експертів фіксуються і після закінчення сеансу аналізуються, тобто висловлені ідеї й рішення групуються та класифікуються за різноманітними ознаками, оцінюється ступінь корисності та можливості реалізації. Через одну-дві доби після проведення сеансу експерти мають оголосити, чи не виникли якісь нові ідеї та розв'язки.

4.2 Методи обробки експертної інформації

Провівши опитування групи експертів, дістають певну інформацію. Наявність як числових даних, так і змістовних висловлювань експертів спонукає до необхідності застосування якісних і кількісних методів обробки результатів групового експертного оцінювання.

Залежно від цілей експертного оцінювання під час обробки результатів опитування виникають такі основні завдання:

1) визначення узгоджених думок (суджень) експертів;
2) побудова узагальненої оцінки об'єктів;
3) визначення залежності між судженнями експертів;
4) визначення відносних ваг об'єктів;
5) оцінювання надійності (ризику) результатів експертизи.

Визначити узгодженість оцінок експертів необхідно для того, щоб підтвердити правильність гіпотези: експерти є досить точними «вимірювачами», а також виявити можливі угруповання в експертній групі. Узгодженість думок експертів оцінюють, обчислюючи кількісну міру, що характеризує ступінь близькості індивідуальних думок.

Завдання **побудови узагальненої оцінки** об'єктів групою експертів на підставі індивідуальних оцінок експертів є однією з проблем у груповому експертному оцінюванні. Якщо експерти оцінювали об'єкти в кількісній шкалі, то завдання побудови групової оцінки полягає у визначенні середнього значення або медіани оцінки. У вимірюванні в порядковій шкалі методом ранжування або ***парного порівняння*** метою обробки індивідуальних оцінок експертів є побудова узагальненого впорядкування об'єктів на підґрунті усереднення оцінок експертів.

Обробка результатів експертизи вручну пов'язана з великими трудовими затратами (навіть у разі розв'язування простих задач впорядкування), через це її доцільно виконувати за допомогою обчислювальної техніки. ***Застосування ЕОМ*** висуває проблему створення відповідних комп'ютерних програм, які реалізують алгоритми обробки результатів експертного оцінювання.

Оцінка узгодженості суджень експертів ґрунтується на використанні поняття компактності, наочне уявлення про яке дає геометрична інтерпретація результатів експертизи. Оцінка кожного експерта подається як точка в деякому просторі з введеним у ньому поняттям відстані. Коли точки, що характеризують оцінки всіх експертів, розміщені на невеликій відстані одна від одної, тобто утворюють компактну групу, то, очевидно, це можна інтерпретувати як добру узгодженість думок експертів. А якщо точки в просторі розкидані на значні відстані, узгодженість думок експертів невисока. Можливо, що точки оцінки експертів розміщені в просторі у такий спосіб, що утворюють одну або кілька компактних груп. У такому разі в експертній групі існують два або кілька відмінних один від одного поглядів на оцінку об'єктів.

Коли використовують кількісні шкали вимірювання та оцінюють лише один параметр об'єкта, всі думки експертів можна подати як точки на числовій осі, розглядаючи їх як реалізацію випадкової величини. Тому для оцінювання центру групування та розкиду точок можна використати добре розроблені методи математичної статистики. Центр групування точок визначають як

математичне сподівання (середнє значення) або як медіану випадкової величини, а розкид кількісно оцінюють дисперсією випадкової величини.

Мірою узгодженості оцінок експертів, тобто компактності розміщення точок на числовій осі, може бути відношення середньоквадратичного відхилення до математичного сподівання випадкової величини.

Для ранжування об'єктів використовують міру узгодженості думок групи експертів — дисперсійний коефіцієнт конкордації (коефіцієнт злагоди).

Поряд з дисперсійним коефіцієнтом конкордації як міру узгодженості суджень експертів використовують ентропійний коефіцієнт конкордації.

Нехай маємо випадок, коли величини $x_{is}^{h},\ s=1,...,d;i=1,...,m,$ знайдено методами безпосереднього оцінювання або послідовного порівняння, тобто вони є числами або балами. Щоб дістати групову оцінку об'єктів, можна скористатись середнім значенням оцінки для кожного об'єкта:

$$x_i = \sum_{h=1}^{I}\sum_{s=1}^{d} q_h k_s x_{is}^{h}, \qquad (1)$$

де q_h — коефіцієнт ваг показників порівняння об'єктів;

k_s — коефіцієнти компетентності експертів.

Ці коефіцієнти є нормованими величинами:

$$\sum_{h=1}^{I} q_h = 1,\ \sum_{s=1}^{d} k_s = 1. \qquad (2)$$

Коефіцієнти ваг показників можна знайти експертно.

Відшукання групової експертної оцінки підсумовуванням індивідуальних оцінок з вагами компетентності та важливості показників у вимірюванні властивостей об'єктів у кількісних шкалах ґрунтується на такому припущенні: виконуються аксіоми теорії корисності фон Неймана—Моргенштерна як для індивідуальних оцінок, так і для групової, а також умови, що об'єкти не різняться щодо всіх індивідуальних оцінок у груповому розумінні (частковий принцип Паретто). У реальних задачах ці умови, як правило, не виконуються, тому на практиці групову оцінку об'єктів часто знаходять підсумовуванням з вагами індивідуальних оцінок експертів.

Коефіцієнти компетентності експертів можна розрахувати за апостеріорними даними, тобто за результатами оцінювання об'єктів. Основною ідеєю цього обчислення є гіпотеза про те, що компетентність експертів оцінюється за ступенем узгодженості їх оцінок з груповою оцінкою об'єктів.

Алгоритм обчислення коефіцієнтів компетентності експертів має вигляд рекурентної процедури:

$$x_i^t = \sum_{s=1}^{d} x_{is} k_s^{t-1}, \;\; i = 1, \;\; ..., \; m; \;\; t = 1, \; 2, \;\; ...; \tag{3}$$

$$\lambda^t = \sum_{i=1}^{m} \sum_{s=1}^{m} x_{is} x_i^t, \;\; t = 1, \; 2, \;\; ...; \tag{4}$$

$$k_s^t = \frac{1}{\lambda^t} \sum_{i=1}^{m} x_{is} x_i^t, \quad s = 1, ..., d; \;\; t = 1, 2, ... \tag{5}$$

Обчислення починають з $t = 1$. У (5) початкові значення коефіцієнтів компетентності беруть однаковими й такими, що дорівнюють $k_s^0 = 1/d$. Тоді групові оцінки об'єктів першого наближення дорівнюють середнім арифметичним значенням оцінок експертів:

$$x_i^1 = \frac{1}{d} \sum_{s=1}^{d} x_{is}, \;\; i = 1, ..., m. \tag{6}$$

Далі за (4) обчислюють

$$\lambda^1 = \sum_{i=1}^{m} \sum_{s=1}^{d} x_{is} x_i^1, \tag{7}$$

а також за (5) — значення коефіцієнтів компетентності першого наближення

$$k_s^1 = \frac{1}{\lambda^1} \sum_{i=1}^{m} x_{is} x_i^1, \;\; s = 1, ..., d. \tag{8}$$

Використовуючи коефіцієнти компетентності першого наближення, можна повторити весь процес обчислення за формулами (3), (4), (5) і дістати другі наближення величин.

Розглянемо випадок, коли експерти вимірюють об'єкти в порядковій шкалі методом ранжування. Щоб спростити міркування, розглянемо спочатку випадок однієї ознаки порівняння, тобто показника величин Y_{ij}. Кожне ранжування можна подати у вигляді матриці **парних порівнянь з елементами**, що визначаються за правилом

$$Y_{ij} = \begin{cases} 2, \text{ якщо об'єкт } i \text{ має переваги над об'єктом } j \; (i > j), \\ 1, \text{ якщо визначено рівність об'єктів } (i = j), \\ 0, \text{ якщо об'єкт } j \text{ має переваги над об'єктом } i \; (i < j). \end{cases}$$

Таблиця 4.1

Результат порівняння заноситься до таблиці-матриці для парного порівняння за ознакою Y_{ij}

Об'єкти, що порівнюються	1	2	3	4	5	Сума балів	Абсолютне рангове місце
1	X	1	1	1	2	5	I
2	1	X	1	1	2	5	II
3	1	1	X	1	2	5	III
4	1	1	1	X	1	4	IV
5	0	0	0	1	X	1	V

Після формування матриці парних порівнянь нами отримується вектор пріоритетів елементів деякого рівня відносно найвищого значення. Обчислена сума $\sum_{j=1}^{n} Y_{ij}$ (за рядком) дозволяє визначити ранг (відносну значимість об'єктів). Той об'єкт, для якого сума буде мати найбільше значення, може бути визнаним із найбільшим рівнем ознаки особистості, що досліджується.

Якщо є *d* експертів, то кожний експерт дає своє ранжування, якому відповідає матриця попарних порівнянь. Отже, кількість матриць попарних порівнянь дорівнює кількості експертів.

Введемо відстань (метрику) між матрицями попарних порівнянь:

$$d_{sj} = \sum_{i,\, k=1}^{m} \left| y_{ik}^{s} - y_{ik}^{j} \right|, \quad s, j = 1, \ldots, d. \tag{10}$$

Зміст цього виразу полягає у тому, що відстань між матрицями попарних порівнянь визначається кількістю позарозрядних незбігів усіх значень елементів матриць (метрика Хеммінга).

Використовуючи цю метрику, визначимо узагальнене ранжування як матрицю попарних порівнянь, що найкраще узгоджується з матрицями попарних порівнянь, одержуваних з ранжувань експертів.

Поняття найкращого узгодження на практиці здебільшого визначають як медіану.

Медіана — це така матриця попарних порівнянь, сума відстаней якої до всіх матриць попарних порівнянь, що одержують експерти, є мінімальною:

$$\| y_{ik}^{*} \| \Leftarrow \min_{y_{ik}} \sum_{s=1}^{d} \sum_{i,k=1}^{m} \left| y_{ik}^{s} - y_{ik} \right|. \qquad (11)$$

Покажемо, що матрицю попарних порівнянь, які відповідають медіані, будують за принципом простої більшості голосів експертів для кожного елемента матриці. Модуль різниці змінних у (11) дорівнює або одиниці, або нулю, тому цей модуль дорівнює своєму квадрату. Отже, замість виразу (11) можна записати

$$\min_{y_{ik}} \sum_{s=1}^{d} \sum_{i,k=1}^{m} \left(y_{ik}^{s} - y_{ik} \right)^2. \qquad (12)$$

Розкриваючи квадрат та враховуючи, що квадрат змінної дорівнює самій змінній (з (9)), з (12) знаходимо

$$\min_{y_{ik}} \sum_{s=1}^{d} \sum_{i,k=1}^{m} \left(y_{ik}^{s} - 2\, y_{ik}^{s}\, y_{ik} + y_{ik} \right). \qquad (13)$$

Введемо позначення

$$a_{ik} = \sum_{s=1}^{d} y_{ik}^{s}. \qquad (14)$$

Зробивши відповідні перетворення у (13) з урахуванням (14) отримаємо

$$\min_{y_{ik}} \sum_{i,k=1}^{m} (a_{ik} - 2a_{ik}\, y_{ik} + y_{ik}\, d) = \min_{y_{ik}} \left[\sum_{i,k=1}^{m} a_{ik} - 2 \sum_{i,k=1}^{m} y_{ik} (a_{ik} - d/2) \right]. \qquad (15)$$

Перша сума в квадратній дужці стала й не залежить від змінної y_{ik}. Тому мінімум виразу в квадратних дужках з (4.15) відповідає максимуму другої суми, тобто

$$\min_{y_{ik}} \sum_{s=1}^{d} \sum_{i,k=1}^{m} \left| y_{ik}^{s} - y_{ik} \right| = \max_{y_{ik}} \sum_{i,k=1}^{m} y_{ik} (a_{ik} - d/2). \qquad (16)$$

Максимум за змінними y_{ik}, що набувають значення 0 або 1, досягається за такої умови

$$y_{ik}^{*} = \begin{cases} 1, \ \text{якщо} \ \ a_{ik} \geq \dfrac{d}{2}; \\ 0, \ \text{якщо} \ \ a_{ik} < \dfrac{d}{2}, \end{cases} \qquad (17)$$

де d — кількість експертів.

Величини a_{ik} згідно з (14) є кількістю голосів, поданих експертами за перевагу *i*-го об'єкта порівняно з *k*-м об'єктом. Отже, всі елементи узагальненої матриці попарних порівнянь визначаються за правилом більшості голосів.

У розглянутому алгоритмі побудови узагальненої матриці попарних порівнянь можна врахувати компетентність експертів, увівши коефіцієнти компетентності k_s у співвідношення:

$$\| y_{ik}^* \| \Leftarrow \min_{y_{ik}} \sum_{s=1}^{d} \sum_{i,k=1}^{m} k_s \left| y_{ik}^s - y_{ik} \right|. \qquad (18)$$

Виконуючи перетворення, аналогічні співвідношенням (12)—(17), дістанемо для випадку врахування коефіцієнтів компетентності експертів таке правило побудови узагальненої матриці попарних порівнянь:

$$y_{ik}^* = \begin{cases} 1, \text{ якщо } b_{ik} \geq \dfrac{1}{2}; \\ 0, \text{ якщо } b_{ik} < \dfrac{1}{2}, \end{cases} \qquad (19)$$

де

$$b_{ik} = \sum_{s=1}^{d} k_s y_{ik}^s, \; i, k = 1, \ldots, m. \qquad (20)$$

Значення порогу в (19) дорівнює 1/2, оскільки значення b_{ik} можна розглядати як імовірність того, що *i*-й об'єкт переважає *k*-й.

За наявності кількох ситуацій експерти впорядковують об'єкти (рішення) для кожної ситуації окремо. Коли відомі ймовірності ситуацій $P_1, P_2, \ldots, P_n$, де *n* — кількість ситуацій, то можна побудувати узагальнене ранжування, усереднене за всіма ситуаціями. Припишемо елементам матриць попарних порівнянь індекс *j* — номер ситуації y_{ikj}^s. Тоді узагальнена матриця попарних порівнянь визначається з умови

$$\| y_{ik}^* \| \Leftarrow \min_{y_{ik}} \sum_{j=1}^{n} \sum_{s=1}^{d} \sum_{i,k=1}^{m} k_s p_j \left| y_{ikj}^s - y_{ikj} \right|. \qquad (21)$$

Виконуючи перетворення, аналогічні попереднім, дістаємо таке правило побудови узагальненої матриці попарних порівнянь, усереднених за допомогою ймовірностей за всіма ситуаціями:

$$y_{ik}^* = \begin{cases} 1, \text{ якщо } c_{ik} \geq \dfrac{1}{2}; \\ 0, \text{ якщо } c_{ik} < \dfrac{1}{2}, \end{cases} \qquad (22)$$

$$c_{ik} = \sum_{j=1}^{n} \sum_{s=1}^{d} p_j k_s y_{ikj}^{s}. \qquad (23)$$

У частковому випадку однакової компетентності експертів

$$c_{ik} = \frac{1}{d} \sum_{j=1}^{n} \sum_{s=1}^{d} p_j y_{ikj}^{s}. \qquad (24)$$

Правило побудови елементів узагальненої матриці попарних порівнянь (22) є найзагальнішим і включає як частковий випадок правила (17) і (19).

Узагальнену матрицю попарних порівнянь можна побудувати, врахувавши умовні ймовірності прийняття помилкових рішень. Алгоритм знаходження елементів цієї матриці має вигляд, аналогічний (22).

Правило (22) визначає групову оцінку попарних порівнянь. Щоб дістати узагальнене ранжування за матрицею попарних порівнянь, застосовують послідовне виокремлення недомінуючих об'єктів. Оскільки матриця попарних порівнянь описує граф, то послідовне виокремлення недомінуючих об'єктів відповідає послідовному виокремленню ядра графа. Для послідовного виокремлення недомінуючих об'єктів виконують операцію транзитивного замикання матриці попарних порівнянь та ранжування об'єктів за цією матрицею на підставі підрахунку кількості одиниць у кожному стовпчику матриці. Об'єкту, що має у своєму стовпчику найменшу кількість одиниць, присвоюють перший ранг; другого рангу набуває об'єкт, що має у своєму стовпчику більше одиниць, ніж перший об'єкт, але менше, ніж усі інші об'єкти, тощо.

Побудова узагальненого ранжування об'єктів за результатами їх попарних порівнянь передбачає, очевидно, що всі об'єкти експерти порівнюють один з одним. Але можлива побудова узагальненого ранжування за результатами попарних порівнянь лише частини об'єкта. Для цього випадку алгоритми побудови узагальненого ранжування мають складніший вигляд.

Під час обробки результатів ранжування можуть виникнути задачі визначення залежності ***між ранжуванням двох експертів*** або між двома ознаками. У цих випадках мірою взамозв'язку може бути ***коефіцієнт рангової кореляції***. Характеристикою взамозв'язку ранжувань або цілей є матриця коефіцієнтів рангової кореляції. Відомі коефіцієнти рангової кореляції Спірмена та Кендалла.

Коефіцієнт рангової кореляції Спірмена:

Аналіз отриманих даних порівняння виконується із застосуванням методу рангової кореляції Спірмена за формулою:

$$rs = 1 - \frac{6 \cdot \sum (X_i - Y_i)^2}{N \cdot (N^2 - 1)}, \qquad (25)$$

де X_i й Y_i - ранги, що визначені відповідно за методом парного порівняння та за показником профілю адаптивності суб'єктів навчання експериментальної та контрольної груп;

N – кількість суб'єктів навчання, що піддаються ранжуванню.

Коефіцієнт рангової кореляції Спірмена змінюється від –1 до +1.

Рівність одиниці досягається за однакових ранжувань, тобто коли $r_{1j} = r_{2j}\,(j=1,...,m)$. Значення $\rho=-1$ відповідає протилежним ранжуванням (пряме та обернене ранжування). У разі рівності коефіцієнтів кореляції нулю ранжування вважаються лінійно незалежними.

Оцінка коефіцієнта кореляції, обчислювана за формулою (25), є випадковою величиною. Щоб визначити значущість цієї оцінки, необхідно задати величину ймовірності, обчислити значення порога, прийняти рішення про значущість коефіцієнта кореляції:

$$\varepsilon = \frac{1}{\sqrt{m-1}}\psi\left(\frac{1-\beta}{2}\right), \qquad (26)$$

де m — кількість об'єктів; $\psi(x)$ — функція, обернена до функції $\varphi(x)=\frac{1}{\sqrt{2\pi}}\int_0^x e^{-\frac{t^2}{2}}dt,$ для якої складено таблиці. Коли обчислено порогове значення, оцінка коефіцієнта кореляції вважається значущою, якщо $|\rho|>\varepsilon$.

Для визначення значущості коефіцієнта кореляції Спірмена можна скористатись критерієм Стьюдента.

Лабораторна робота № 3
Емпіричні методи дослідження. Експертні процедури та методи суб'єктивних оцінок (експертне оцінювання) складних систем

Мета заняття: навчити студентів здійснювати експертні процедури та методи суб'єктивних оцінок.

Послідовність виконання:

1. Ознайомитись із теоретичними відомостями.
2. Проаналізувати та сформувати об'єкт для експертного оцінювання.
3. Оцінити характеристику об'єкта методом попарних порівнянь.
4. Проаналізувати отримані дані порівняння двох експертів із застосуванням методу рангової кореляції Спірмена.
5. Проаналізувати отримані дані порівняння п'яти експертів обрахувавши дисперсійний коефіцієнт конкордації.

6. Оформити звіт лабораторної роботи.

Приклад розрахунків.

Приклад 1:

Таблиця 4.2

Оцінювання експертом Х, за методом парного порівняння, деякої характеристики групи співробітників

Прізвище	Бадикін	Бабкін	Богданова	Грудік	Макогон	Пархоменко	Пащенко	Скляр	Ціндин	Сума балів
Бадикін	X	1	0	1	0	1	0	0	1	4
Бабкін	1	X	0	1	0	0	0	1	1	4
Богданова	2	2	X	2	0	0	0	1	1	8
Грудік	1	1	0	X	0	0	1	0	0	3
Макогон	2	2	2	2	X	0	1	1	1	11
Пархоменко	1	2	2	2	2	X	1	1	2	13
Пащенко	2	2	2	1	1	1	X	1	2	12
Скляр	2	1	1	2	1	1	1	X	1	10
Ціндин	1	1	1	2	1	0	0	1	X	7

Таблиця 4.3

Обрахунку персентилю за методом парного порівняння, деякої характеристики групи співробітників

Прізвище	СУМ БАЛ	Абсол Ранг місце	Віднос ранг	% Ранговий показник PR (персентиль)
Грудік	3	1	1	5,555555556
Бадикін	4	2	2,5	22,22222222
Бабкін	4	3	2,5	22,22222222
Ціндин	7	4	4	38,88888889
Богданова	8	5	5	50
Скляр	10	6	6	61,11111111
Макогон	11	7	7	72,22222222
Пащенко	12	8	8	83,33333333
Пархоменко	13	9	9	94,44444444

Таблиця 4.4

Розрахунку коефіцієнту рангової кореляції Спірмена оцінок двох експертів (значення X та Y) деякої характеристики групи співробітників

Прізвище	Ранг Xi	Ранг Yi	Xi -Yi	$(Xi - Yi)^2$	$rs = 1 - \frac{6 \cdot \sum (X_i - Y_i)^2}{N \cdot (N^2 - 1)}$
Грудік	1	2	-1	1	0,896
Бадикін	2,5	1	1,5	2,25	
Бабкін	2,5	3,5	-1	1	
Ціндин	4	5,5	-1,5	2,25	
Богданова	5	3,5	1,5	2,25	
Скляр	6	5,5	0,5	0,25	
Макогон	7	8,5	-1,5	2,25	
Пащенко	8	7	1	1	
Пархоменко	9	8,5	0,5	0,25	
Сума	45	45	0	12,5	

Приклад 2:

Визначимо конкордацію експертного оцінювання шести об'єктів **m** п'ятьма експертами **d**.

Таблиця 4.5

Результати ранжування шести об'єктів (X1, X2, X3, X4, X5 и X6) п'ятьма експертами (s1, s2, s3, s4, s5)

Об'єкти	Експерти				
	s_1	s_2	s_3	s_4	s_5
X_1	1	2	1,5	1	2
X_2	2,5	2	1,5	2,5	1
X_3	2,5	2	3	2,5	3
X_4	4	5	4,5	4,5	4
X_5	5	4	4,5	4,5	5,5
X_6	6	6	5	5	5,5

Обчислюємо коефіцієнт конкордації та оцінимо його значущість.

Середнє значення оцінки математичного очікування (середній ранг) дорівнює:

$$\bar{r} = \frac{1}{m}\sum_{i=1}^{m}\sum_{s=1}^{d} r_{is} = 17{,}5$$

.

Величина **S** відповідно дорівнюватиме

$$S = \sum_{i=1}^{m}\left(\sum_{s=1}^{d} r_{is} - \bar{r}\right)^2,$$

$$S = \sum_{i=1}^{6}\left(\sum_{s=1}^{5} r_{is} - 17{,}5\right)^2 = 361$$

.

.Оскільки в ранжуванні є зв'язні ранги, то обчислення коефіцієнта конкордації здійснюється за формулою)

$$W = \frac{12S}{d^2\left(m^3 - m\right) - d\sum_{s=1}^{d} T_s} \quad (27)$$

де

$$T_s = \sum_{k=1}^{H_s}\left(h_k^3 - h_k\right) \quad (28)$$

У даному прикладі приймемо, що тільки в ранжуванні експерта **s1** є одна група зв'язних рангів. У такому випадку H1=1. А в цій групі ми маємо два зв'язних ранги, які дорівнюють по 2,5. Тому k=1 та h1=2. Отже,

$$\mathbf{T_1 = 2^3 - 2 = 6.}$$

Аналогічно обчислюємо $T_2...T_5$:

$\mathbf{T_2=3^3-3=24;}$

$\mathbf{T_3=2^3-2+2^3-2=12;}$

$\mathbf{T_4=2^3-2+2^3-2=12;}$

$\mathbf{T_5=2^3-2=6.}$

Підставивши значення **Ts**, **S** та **m = 6**, **d = 5** у формулу (27) виконуємо обчислення

$$\mathbf{W = 12*361 / [5^2 (6^3-6) - 5*60] = 0{,}874.}$$

Оцінимо значущість коефіцієнту конкордації.

У даному випадку число степенів вільності **n = m - 1 = 6 - 1= 5**.

Тоді за формулою

$$\chi^2 = 12S \Big/ \left[dm(m+1) - \frac{1}{m-1} \sum_{s=1}^{d} T_s \right]$$

$$\chi^2 = 12*361 / [5*6*7 - 0,2*60] = 21,8.$$

Таблиця 4.6

Табличне значення χ^2 для n = 5 при 5% рівня значущості дорівнює 11,07.

Значення χ^2 в залежності від ν та p^*

n	*p*						
	0,3	0,2	0,1	0,05	0,025	0,01	0,005
1	1,07	1,64	2,71	3,84	5,02	6,63	7,88
2	2,41	3,22	4,61	5,99	7,38	9,21	10,6
3	3,67	4,64	6,25	7,81	9,35	11,3	12,8
4	4,88	5,99	7,78	9,49	11,07	13,3	14,9
5	6,06	7,29	9,24	11,07	12,8	15,1	16,7
6	7,23	8,56	10,6	12,6	14,4	16,8	18,5
7	8,38	9,80	12,0	14,1	16,0	18,5	20,3
8	9,52	11,0	13,4	15,5	17,5	20,1	22,0
9	10,7	12,2	14,7	16,9	19,0	21,7	23,6
10	11,8	13,4	16,0	18,3	20,5	23,2	25,2
11	12,9	14,6	17,3	19,7	21,9	24,7	26,8
12	14,0	15,8	18,5	21,0	23,3	26,2	28,3
13	15,1	17,0	19,8	22,4	24,7	27,7	29,8
14	16,2	18,2	21,1	23,7	26,1	29,1	31,3
15	17,3	19,3	22,3	25,0	27,5	30,6	32,8
16	18,4	20,5	23,5	26,3	28,8	32,0	34,3
18	20,6	22,8	26,0	28,9	31,5	34,8	37,2
20	22,8	25,0	28,4	31,4	34,2	37,6	40,0
22	24,9	27,3	30,8	33,9	36,8	40,3	42,8
24	27,1	29,6	33,2	36,4	39,4	43,0	45,6
26	29,2	31,8	35,6	38,9	41,9	45,6	48,3
28	31,4	34,0	37,9	41,3	44,5	48,3	51,0
30	33,5	36,3	40,3	43,8	47,0	50,9	53,7

ν – кількість ступенів вільності;

p – ймовірність того, що χ^2 прийме значення більше ніж у таблиці.

Оскільки **11,07< 21,8**, то гіпотеза про збігання оцінок ранжування експертів приймається.

Тема 5 Автоматизація опрацювання дослідних даних

5.1 Проблеми побудови оптимізаційних моделей в системному аналізі.

Побудова оптимізаційних моделей з математичної точки зору передбачає використання дескриптивних моделей у такому складі: модель (моделі) критеріїв, моделі обмежень. Така постановка дозволяє знайти оптимальні дії для ефективного функціонування системи, або Паретто-оптимальну множину оптимальних дій (у випадку великої кількості критеріїв).

Оптимізаційна модель вимагає змістовного опису і будується у такій послідовності:

1. Розробка дескриптивної моделі, яка описує якість функціонування системи та визначає мету.
2. Побудова множини критеріїв оцінки якості та визначення шкал їх вимірювання.
3. Проведення статистичних досліджень і узгодження з ними моделі.
4. Формалізація задач знаходження оптимальних розв'язків.
5. Вибір методу розв'язування оптимізаційної моделі
6. Оцінювання результатів розв'язування оптимізаційної задачі та їх інтерпретація.

При дослідженні складних систем поняття оптимальності набуває дещо іншого трактування, ніж це прийнято в математиці. В цих умовах проблема вибору оптимальної альтернативи полягає, по-перше – в наявності великої кількості критеріїв та необхідності урахування їх різної важливості, а по-друге в нечіткості формулювання мети, що призводить до нечіткого, описового формулювання критеріїв.

5.2 Постановка завдання інтерполяції

Нехай на відрізку [a, b] задані n+1 точки $x_0, x_1, ...x_n$, які називаються вузлами інтерполяції, і значення деякої функції $f(x)$ в цих точках $f(x_0) = y_0,\ f(x_1) = y_1, ...f(x_n) = y_n$. Треба побудувати функцію $F(x)$ (інтерполююча функція), що належить відомому класу і що має у вузлах інтерполяції ті ж значення, що і $f(x)$, тобто таку, що $F(x_0) = y_0,\ F(x_1) = y_1, ...F(x_n) = y_n$.

Геометрично це означає, що треба знайти криву $y = F(x)$ деякого певного типу, що проходить через задану систему точок $M(x_i, y_i), i = \overline{1,n}$.

Отриману інтерполяційну формулу $y = F(x)$ зазвичай використовують для наближеного обчислення значень певної функції $f(x)$ для значень аргументу x,

відмінних від вузлів інтерполяції. Така операція називається інтерполяцією функції $f(x)$.

У такій загальній постановці завдання може мати незліченну безліч рішень або зовсім не мати їх. Проте це завдання стає однозначним, якщо замість довільної функції $F(x)$ шукати поліном $L_n(x)$ із степенем не вище n, що задовольняє умовам (1), тобто такий, що $L_n(x_0) = y_0, L_n(x_1) = y_1, ...L_n(x_n) = y_n$.

5.2.1 Інтерполяційний багаточлен Лагранжа

Якщо вузли $x_i, i = \overline{1,n}$ різні, то існує єдиний інтерполяційний багаточлен $L_n(x)$ із степенем n. Його можна записувати в різних формах. Розглянемо інтерполяційний багаточлен Лагранжа і Ньютона. У формі Лагранжа інтерполяційний багаточлен $L_n(x)$ має вигляд

$$L_n(x) = \sum_{i=1}^{n} f(x_1) \cdot \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \qquad (1)$$

Наприклад,

$$L_3(x) = f(x_0)\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} + f(x_1)\frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} +$$

$$+ f(x_2)\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} + f(x_3)\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}.$$

4.2.2 Розділені різниці. Інтерполяційний багаточлен Ньютона.

Щоб ознайомитися з інтерполяційним багаточленом у формі Ньютона, введемо в розгляд поняття розділена різниця. Значення $f(x_i), i = \overline{0,n}$ функції $f(x)$ у вузлах називаються розділеними різницями нульового порядку. Числа виду $f(x_i; x_k) = \dfrac{f(x_k) - f(x_i)}{x_k - x_i}$, $k \neq i$ називаються розділеними різницями першого порядку.

Розділена різниця n-го порядку визначається через розділені різниці $n-1$-го порядку по реккурентной формулі

$$f(x_0; x_1; ...; x_n) = \frac{f(x_1; x_2; ...; x_n) - f(x_0; x_1; ...; x_{n-1})}{x_n - x_0}.$$

Розділену різницю k-го порядку ($k \leq n$) можна виразити через $f(x_1)$ по наступній формулі:

$$f(x_0; x_1; ...; x_n) = \sum_{i=0}^{n} \frac{f(x_i)}{\prod_{\substack{j=0 \\ j \neq 1}}^{n} (x_i - x_j)} \quad .$$

Наприклад,

$$f(x_0;x_1;x_2)=\frac{f(x_0)}{(x_0-x_1)(x_0-x_2)}+\frac{f(x_1)}{(x_1-x_0)(x_1-x_2)}+\frac{f(x_2)}{(x_2-x_0)(x_2-x_1)}.$$

Тобто, розділена різниця - симетрична функція своїх аргументів. За допомогою розділених різниць інтерполяційний багаточлен $L_n(x)$ у формі Ньютона можна записати таким чином:

$$L_n(x)=f(x_0)+f(x_0;x_1)(x-x_0)+...+f(x_0;x_1;...;x_n)(x-x_0)(x-x_1)...(x-x_{n-1}) \qquad (2)$$

5.2.3 Похибка інтерполяції

Нехай $L_n(x)$ - інтерполяційний багаточлен, побудований для функції $f(x)$ по вузлах інтерполяції $x_0,x_1,...x_n$ з відрізку $[a,b]$ по формулі (1) або (2). Тоді можна написати наближену рівність $f(x)\approx L_n(x)$.

Різниця $f(x)-L_n(x)$, що виражає похибку інтерполяції, називається залишковим членом і позначається $R_n(x)$. Його можна записати у вигляді

$$R_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\cdot\omega_{n+1}(x), \qquad (3)$$

де $\omega_{n+1}(x)=(x-x_0)...(x-x_n)$, а $\xi=\xi(x)\in(a,b)$ - деяка невідома точка, або за допомогою розділених різниць у виді

$$R_n(x)=f(x;x_0;x_1;...x_n)\cdot\omega_{n+1}(x) \qquad (4)$$

Залишковий член $R_n(x)$ у формі (3) використовують у тому випадку, коли $f(x)\in C^{n+1}[a,b]$. В цьому випадку

$$|R_n(x)|\le\frac{M_{n+1}}{(n+1)!}\cdot|\omega_{n+1}(x)|, \qquad (5)$$

де $M_{n+1}\ge\max\limits_{x\in[a,b]}|f^{(n+1)}(x)|$.

Якщо ж $f(x)$ не належить $C^{n+1}[a,b]$, то користуються залишковим членом у формі (4).

Приклад. З якою точністю можна вичислити $\sqrt{115}$ за допомогою інтерполяційної формули Лагранжа (1) для функції $y=\sqrt{x}$, вибравши вузли інтерполяції 100, 121, 144?

Рішення. Маємо $y'=\frac{1}{2}x^{-1/2}$, $y''=-\frac{1}{4}x^{-3/2}$, $y'''=\frac{3}{8}x^{-5/2}$.

Звідси $M_3=\max|y'''(x)|=\frac{3}{8}10^{-5}$ при $100\le x\le144$. На підставі формули (5), отримуємо $|R_2|\le\frac{3}{8}10^{-5}|(115-100)(115-121)(115-144)|\le1\cdot6\cdot10^{-3}$.

5.3 Автоматизація опрацювання дослідних даних

5.3.1 Обчислення з використанням чисельних методів

Для визначення коефіцієнтів **An** поліноміальної функції застосовуємо алгоритм, який побудовано на принципі обчислення функції засобами обчислювальних методів.

Процес апроксимації даних розглядається як розв'язок системи рівнянь, які утворюють систему:

$$\begin{cases} Y1 = A0 + A1X1 + A2X1^2 + \ldots + AmX1^m \\ Y2 = A0 + A1X2 + A2X2^2 + \ldots + AmX2^m \\ \quad\quad\quad \ldots \\ Yn = A0 + A1Xn + A2Xn^2 + \ldots + AmXn^m \end{cases}$$

Розв'язок системи рівнянь здійснюємо застосувавши LU-перетворення. Матриця **A,** яка утворилась з коефіцієнтів, подається у вигляді добутку двох матриць L та U.

$$L = \begin{pmatrix} 1 & 0 & 0 & \ldots \\ L_{21} & 1 & 0 & \ldots \\ L_{31} & L_{32} & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}, \quad U = \begin{pmatrix} 1 & U_{12} & U_{13} & \ldots \\ 0 & 1 & U_{23} & \ldots \\ 0 & 0 & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}.$$

Прямий хід за даним алгоритмом – це розв'язання системи

$$U*x = b$$

$$A = \begin{pmatrix} 8 & 1 & 2 \\ 16 & 11 & 7 \\ 24 & 48 & 28 \end{pmatrix}$$

Перший рядок **U** матриці співпадає з першим рядком матриці **A**

$$u_{1j} = a_{1j} \qquad j = 1\ldots n$$

Обрахувати елементи матриці LU можна таким чином (всі кроки виконуються у обов'язковій послідовності, тому що наступні елементи розраховуються з використанням попередніх):

$$L_{j1} = \frac{a_{j1}}{u_{11}}; \qquad j = 2\ldots n \quad (U_{11} \neq 0)$$

Для $i = 2\ldots n$

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \qquad j = i\ldots n$$

$$l_{ji} = \frac{1}{u_{ii}}\left(a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki}\right) \qquad j = i\ldots n$$

для

$$U = \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{pmatrix}$$

$$L_{i1} = \frac{a_{i1}}{a_{11}} = \frac{16}{8} = 2;$$

$U_{22} = 11 - 2*1 = 9$:

$U_{23} = 7 - 2*2 = 3;$

$U_{33} = 28 - 3*2 - 5*3 = 7;$

$L_{32} = \frac{48 - 3*1}{9} = 5;$

отримуємо

$$U = \begin{pmatrix} 8 & 1 & 2 \\ 0 & 9 & 3 \\ 0 & 0 & 7 \end{pmatrix}$$

// код за L U алгоритмом

```
for i:=2 to m+1 do hth[i,1]:= hth [i,1] / hth [1,1];
```

// $L_{j1} = \frac{a_{j1}}{u_{11}};$

```
for i:=2 to m+1 do
        for j:=2 to m+1 do
begin   if  i>j  then   begin
```

// $u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$

```
        z := hth [j,j];
        p:=j-1;
      end
  else   begin
      z:=1;
        p:=i-1;
```

// $l_{ji} = \frac{1}{u_{ii}} \left(a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki} \right)$

```
        end;
    s:=0;
        for k:=1 to p do  s :=  s + hth [i,k] * hth [k,j];
      hth [i,j] :=  (hth [i,j] - s)  /  z;
end;
```

Виконавши LU-перетворення обчислюємо корені рівнянь, застосувавши метод розрахунку за схемою Горнера:

P(X)= (X-Z)(B0 + B1X1 + B2X1^2 +…+ Bm-1X1^{m-1}) +Bm.

5.3.2 Обчислення значень поліномів. Схема горнера

При однократному обчисленні значень полінома невисокої степені послідовність виконання операцій не має особливого значення. Однак, для обчислення поліномів досить високої степені або для обчислення багатьох

значень полінома при різних значеннях аргументу послідовність виконання операцій є суттєвим чинником.

Попереднє обчислення усіх потрібних степенів аргументу x^2, x^3 . . . зазвичай є невигідним, бо потребує досить значної кількості операцій: при обчисленні значень полінома n-го степеню для одержання степенів до x^n включно потрібно $n-1$ множень. Окрім того, потрібні ще n множень на коефіцієнти, тобто усього $2n-1$ множень і n додавань.

Меншої кількості дій - n множень і n додавань - потребує обчислення полінома за так званою *схемою Горнера*.

Нехай даний поліном n-го степеня

$$P_n(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x + a_0 .$$

Запишемо його у вигляді

$$P_n(x) = (((a_n \cdot x + a_{n-1}) \cdot x + a_{n-2}) \cdot x + \ldots + a_1) \cdot x + a_0 .$$

Обчислення значення полінома називають *схемою Горнера*.

Для поліномів загального виду *неможливо побудувати схему більш економну у розумінні кількості операцій*.

Зобразимо схему алгоритму обчислень значень полінома за схемою Горнера. Попередньо введемо позначення

$$A(1) = a_n;\ A(2) = a_{n-1}; \ldots A(k) = a_{n-k+1};\ \ldots\ A(n+1) = a_0, \qquad (10)$$

тобто подамо коефіцієнти у вигляді деякого одновимірного масиву (вектора) із кількістю елементів. Позначимо через значення послідовно виконуваних арифметичних виразів у (9).

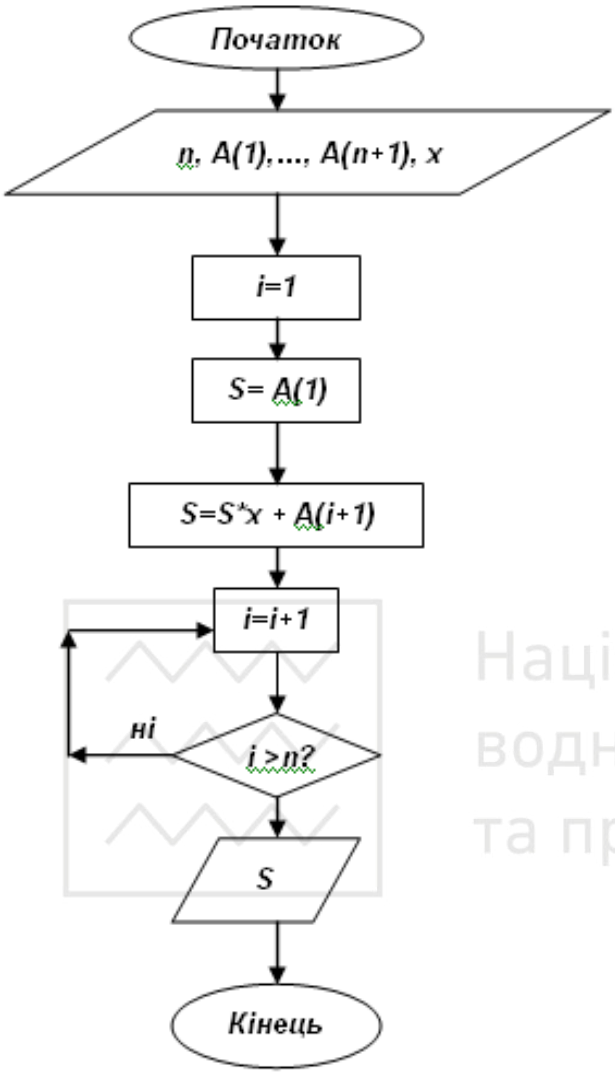
Обчислення поліному **P(x)** в точці **X** за схемою Горнера виконується починаючи від самих внутрішніх дужок і далі у відповідності за (9)

Послідовність дій при обчисленні визначається лапками: спочатку додаємо вміст внутрішньої пари в лапках, далі перемножуємо та додаємо всередині наступної пари лапок і так далі.

Розглянемо приклад

$a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x+a_0 =(((a_5x+a_4)x+a_3)x+a_2)x+a_1)x+a_0$

Для лівої частини	Для правої частини
$\mathbf{x^2}$ $\mathbf{x^3}$ $\mathbf{x^4}$ $\mathbf{x^5}$ $\mathbf{ax}$ $\mathbf{a_2x^2}$ $\mathbf{a_3x^3}$ $\mathbf{a_4x^4}$ $\mathbf{a_5x^5}$	$\mathbf{a_5x}$ $\mathbf{a_5x+a_4}$ $\mathbf{(a_5x+a_4)x}$ $\mathbf{(a_5x+a_4)x+a_3}$ $\mathbf{((a_5x+a_4)x+a_3)x}$ $\mathbf{((a_5x+a_4)x+a_3)x+a_2}$ $\mathbf{(((a_5x+a_4)x+a_3)x+a_2)x}$ $\mathbf{(((a_5x+a_4)x+a_3)x+a_2)x+a_1}$ $\mathbf{((((a_5x+a_4)x+a_3)x+a_2)x+a_1)x}$ $\mathbf{((((a_5x+a_4)x+a_3)x+a_2)x+a_1)x+a_0}$
9 операцій множення та 6 операцій додавання ***Разом 15 операцій***	5 операцій множення та 5 операцій додавання ***Разом 10 операцій*** **(N множення та N додавання)**



Алгоритм схеми Горнера

вхід:

n- ціле,

A[n+1] - дійсне,

x - дійсне

// n - степінь полінома

// A[n+1] - масив коефіцієнтів по

// зменшенню степені

// потрібно обчислити значення для значення X

Початок обчислення

S := A0; // Ініціалізація значення поліному

// цикл для i від 0 до n

S := S * x + A[i]; // Обчислення нового значення

// при досягненні n кінець циклу

// виведення S

Лабораторна робота № 4
Автоматизація опрацювання дослідних даних

Мета заняття: навчити студентів опрацьовувати за допомогою обчислювальної техніки отримані дослідні данні у формі поліному.

Послідовність виконання:

1. Ознайомитись із теоретичними відомостями.
2. Обчислити коефіцієнти поліному у формі Лагранжа і у формі Ньютона.
3. Розробити програму обчислення коефіцієнтів поліному.
4. Оформити звіт лабораторної роботи.

Використана література

1. Згуровський М.З. Системный анализ в исследовании сложных физических процесов и полей / М.З.Згуровський и др. – К. : АН Украины, 1993. – 37 с.
2. Архангельский А.Я. Приемы программирования в Borland C++. – М.:ООО “БИНОМ-ПРЕСС”, 2003.– 784 с.
3. Цветков Э.И. Основы теории статистических решений / Э.И. Цветков. – Л.: Энергия, 1989. – 288 с.
4. www.citmgu.ru/Borland C++ .