МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

МІЖНАРОДНИЙ КЛАСИЧНИЙ УНІВЕРСИТЕТ

ІМЕНІ ПИЛИПА ОРЛИКА

Економічно-технологічний факультет

Кафедра інженерних технологій

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського рівня) вищої освіти

на тему:

«**Математичне моделювання температурної динаміки в процесі теплового забруднення територій**»

Зі спеціальності 123

«КОМП’ЮТЕРНА ІНЖЕНЕРІЯ»

Виконавець:

Пономарьов О.І.

Науковий керівник:

к.т.н., доц. Гайша О.О.

Миколаїв – 2025

РЕФЕРАТ

Текстова частина магістерської роботи: 75 с., 18 рис., 1 табл., 5 додатків, 19 джерел.

ТЕПЛОВЕ ЗАБРУДНЕННЯ, МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ, ПРОГНОЗУВАННЯ, РЕГРЕСІЙНИЙ АНАЛІЗ, КВАЛІМЕТРІЯ, JAVASCRIPT

Об’єкт дослідження – процес розповсюдження теплового забруднення у часі.

Мета роботи – підвищення точності математичного моделювання процесу розповсюдження теплового забруднення у часі.

Метод дослідження – аналітичний з використанням комп’ютерних технологій.

В роботі створена математична модель для прогнозування результатів теплового забруднення місцевості, тобто перевищення температурних показників міської місцевості тих значень, що спостерігаються для аналогічних по клімату заповідних місць. Теплове забруднення на сьогоднішній день є найнебезпечнішим, адже загрожує людській цивілізації в цілому за рахунок глобальної зміни клімату планети, тому суттєвий інтерес представляє моделювання його розповсюдження у часі, яке виконується для конкретних заданих місць (розповсюдження його у просторі не представляє наукового інтересу, оскільки осередки теплового забруднення є очевидними і співпадають із сучасними мегаполісами та агломераціями). Для вирішення поставленої задачі вперше виконано синтез статистичного підходу на базі регресійного аналізу та кваліметричного оцінювання об’єкту (місцевості). Розроблено веб-додаток, що здійснює відповідні розрахунки на довільному комп’ютері, для якого існує програма-браузер (кросплатформенне рішення).

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – База даних
ГДК – Гранично допустима концентрація
ГМО – Генно-модифіковані організми
ІАД – Інтелектуальний аналіз даних
ІТ – Інформаційні технології
ОО – Об’єктно-орієнтований (-на, -не)
ООП – Об’єктно-орієнтоване програмування
ПЗ – Програмне забезпечення
ПК – Персональний комп’ютер
СУБД – Система управління базами даних
ТНМ – Теорія нечітких множин
ШНМ – Штучна нейронна мережа
AJAX – Asynchronous Javascript And XML
ASP – Active Server Pages
CSS – Cascading Style Sheet
CSV – Comma Separated Values
FIFO – First Input First Output
GUI – Graphics User Interface
HTML – HyperText Markup Language
JSP – Java Server Pages
PC – Personal Computer
PHP – Personal Home Page, **P**HP **H**ypertext **P**reprocessor
SQL – Structured Queries Language
UML – Unified Modeling Language
WWW – World Wide Web
XML – eXtensible Markup Language

ВСТУП

Незважаючи на бурхливий розвиток технологій, людина як була кілька тисяч років тому назад, так і залишається досить ніжним створінням, яке для свого нормального існування потребує цілого комплексу безпечних, а краще, комфортних умов життя. В першу чергу, виставляються вимоги до (по мірі зменшенні нагальності, оперативності забезпечення):

– чистоти повітря, яке вона вдихає;

– чистоти води, яка використовується для пиття;

– біологічної чистоти рослин та тварин, які використовуються у їжу.

Менш очевидним, але безперечно необхідним є контроль наступних параметрів:

– хімічного складу ґрунтів, які використовуються для сільськогосподарських цілей;

– чистоти підземних, ґрунтових вод;

– радіаційної чистоти навколишнього середовища.

Насправді ж найбільш небезпечним, але і найбільш незвичайним, є зовсім інший тип забруднення, і ми із впевненістю можемо стверджувати, що це теплове (!) забруднення місцевості. На перший погляд немає нічого страшного у виділенні додаткових кількостей теплоти, адже будь-який живий організм сам по собі є джерелом тепла. Але у сучасному людському світі мова іде уже не про сотні ватт теплової енергії, що виділяються за рахунок процесів життєдіяльності організму людини. Кожна людина зараз власноруч використовує приготування/підігрівання їжі та напоїв (електричним струмом чи одним із типів палива), опалення у зимовий період, використання довільної техніки (яка за законами термодинаміки уся є джерелом побічного тепла), тощо. Крім цього, не слід забувати про таке потужне джерело тепла, як промислове виробництво, масштаби якого також напряму залежать від кількості населення, на обслуговування якого направлене дане виробництво. Споживачами будь-якого промислового виробництва «у середньому» є усі люди, для яких його продукція географічно є досяжною (зокрема, часто кількість споживачів прирівнюють до кількості населення країни, або великого регіону – за умови його високої усамітненості). Саме тому часто розглядають промислове виробництво «на душу населення», і саме тому

абсолютно законно можна сказати, що зі збільшенням кількості населення пропорційно зростає теплове генерування у відповідному регіоні.

Таким чином, теплове забруднення (цим терміном називатимемо перевищення над фоновим (природним) рівнем температурних показників місцевості, на якій активно ведеться життєдіяльність людини, або супутні процеси типу виробництва і т.п.), на відміну від інших видів забруднення, що діють локально, здійснюється системно і одночасно по всьому світу, по всім місцям суттєвого скупчення людей та їхніх промислових потужностей (в основному це все скупчено в межах великих міст). Така безперервна і постійна дія є однією з причин глобального потепління, реальність якого не заперечує вже жоден із серйозних вчених.

Таким чином, суттєвий інтерес представляє прогнозування теплового забруднення, тобто на скільки градусів у майбутньому температурні параметри у містах (головні джерела теплового забруднення) перевищуватимуть фонові значення (що спостерігаються у схожих місцевостях на аналогічних широтах, але за відсутності міських поселень).

Важливою особливістю теплового забруднення (як і інших системних, глобальних типів забруднення типу накопичення пластику у воді, домішок у повітрі, тощо) є те, що його доцільніше розглядати не у просторі (таке розповсюдження є важливим при виникненні аварій тимчасового характеру, наслідки яких поступово згасають, як, наприклад, при аварії на Чорнобильській АЕС у перші дні надважливою була інформація про напрямок руху радіоактивної хмари, тобто її розповсюдження у просторі), а у часі, задаючи тоді різні точки, де ведеться спостереження, а потім виконується передбачення. Дійсно, усі знають, що у Світовому океані ширяться мікрочастинки пластику, але більший інтерес представляє не їх просторове поширення (адже вони є практично усюди), а динаміка наростання їх концентрації у деяких стандартних точках водозабору, реперних точках (тобто важливе розповсюдження забруднення у часі). Для теплового забруднення питання просторового поширення взагалі не є актуальним, оскільки його осередки очевидно співпадають з сучасними агломераціями та мегаполісами. Розповсюдження ж теплового забруднення у часі (в заданих точках, містах) навпаки представляє суттєвий інтерес і його моделювання є дуже доцільним.

У такій постановці описана проблема являє собою задачу прогнозування, і, відповідно, може вирішуватися різними шляхами. Таким

чином, моделювання теплового забруднення з метою визначення його прогнозованих значень у майбутньому є важливою і актуальною задачею на стику сучасної математики та галузі ІТ (інформаційних технологій).

Метою роботи є підвищення точності математичного моделювання процесу розповсюдження теплового забруднення у часі.

Для досягнення поставленої мети слід вирішити наступні задачі дослідження:

– проаналізувати описані у сучасній науково-технічній літературі методи моделювання процесів поширення забруднень, зокрема, теплових;

– вибрати метод, у який можна ввести покращення, для підвищення його ефективності;

– здійснити математичне обґрунтування запропонованого покращення;

– виконати програмну реалізацію запропонованого методу;

– зробити висновки по роботі та окреслити її перспективи розвитку.

Об’єкт дослідження – процес розповсюдження теплового забруднення у часі.

Предмет дослідження – методи моделювання динаміки (у часі) процесу теплового забруднення.

Наукова новизна отриманих результатів полягає у розробці оригінального методу моделювання процесу теплового забруднення, який поєднує статистичні підходи (на основі виділення часових трендів зміни контрольованої величини) та кваліметричні або скорингові способи (отримання результату шляхом нарахування балів за окремі наявні показники та об’єднання окремих показників), що дозволяє підвищити точність (прогнозну силу) відповідної моделі.

Практичне значення роботи полягає у розробці програмного продукту, за допомогою якого можна наочно спостерігати за результатами моделювання процесу розповсюдження теплових забруднень у часі.

Методи досліджень, що застосовуються у роботі: методи математичної статистики, методи кваліметрії, методи технологій програмування.

У перспективі дослідження може розширюватися «в ширину» (шляхом залучення нових міст) та «вглиб» (шляхом удосконалення самої моделі). У першому випадку розширюється сфера застосування програми, у другому – покращується точність прогнозування.

Результати дослідження опубліковані у трьох наукових публікаціях (2 – у матеріалах Міжнародних конференцій, 1 – у науковому журналі).

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Види забруднень навколишнього середовища та доцільність їх моделювання за допомогою програмного забезпечення спеціальної розробки

Безпека життя і здоров’я людини є однією з найвищих цінностей суспільства в сучасному світі. Нажаль, існує чимало факторів, що загрожують цим беззаперечним важливим складовим нашого життя. Цілий багатогранний комплекс утворюють різноманітні уражаючі фактори, пов’язані із забрудненням.

Забруднення є багатоплановим процесом і в галузі безпеки життєдіяльності та охорони праці виділяють наступні різні його типи:

а) хімічне – при неконтрольованому поширенні хімічної речовини у:

1) газоподібній фазі (забруднення повітря, яке може передаватися за рахунок інтенсивної у газах дифузії та конвективних потоків);

2) рідкій фазі (у 99% випадків несучою рідиною є вода; в цьому випадку також основну роль грають дифузія та конвекція, хоча не виключений варіант чисто механічного перенесення рідкого забруднювача з одного місця на інше за допомогою ще одного якогось носія);

3) твердому стані – за рахунок механічного перенесення (силою повітря, як при ураганах, води – як при повені, механічним перенесенням на якомусь носії).

б) фізичне – при розповсюдженні підвищених значень (у порівнянні із незбуреним фоном) різних фізичних величин. Для цього випадку можна виділити наступні підвиди:

1) радіаційне (виникає при поширенні у навколишньому середовищі радіоактивної речовини, що, відповідно, підлягає одному із трьох варіантів розпаду: α, β або γ; надзвичайно небезпечне);

2) теплове (зважаючи, що більшість об’єктів оточуючого середовища мають суттєві теплоємності, цей вид забруднення досить довго не зникає навіть при повному зникненні джерел забруднення; крім того, нагріті маси повітря, води можуть переміщуватися на порівняно великі відстані);

3) шумове забруднення (його специфіка полягає у дуже малій інерційності: при зникненні джерела воно також миттєво пропадає, тому говорити у цьому випадку про стійке забруднення немає сенсу; крім того шумове забруднення не може поширюватися самовільно на невизначені відстані від джерела);

4) магнітне (також чітко прив'язане до джерела і пропадає при прибиранні самих магнітів, самовільно не поширюється);

5) електростатичне (при переміщенні статичних зарядів може змінювати свою конфігурацію, але зазвичай, у невеликих межах);

6) світлове (аналогічно неінерційне, самовільно не поширюється);

7) вібраційне (аналогічно).

в) біологічне забруднення – не тільки може поширюватися далеко від джерела, а й самовільно підвищувати власну інтенсивність, причому у досить широких межах (яка може характеризуватися, наприклад, масою живої речовини на одиницю площі);

г) механічне забруднення великогабаритними відходами життєдіяльності людини (як-то при організації сміттєзвалищ). Характеризується тим, що самовільному поширенню не підлягає, може зростати лише контрольовано, «під наглядом» людини.

Аналіз особливостей усіх типів забруднень показує, що деякі з них підлягають самовільному поширенню, до того ж являються суттєвими небезпечними та шкідливими факторами. Так, особливу увагу слід приділяти питанням двох класів:

– хімічного, радіаційного та біологічного забруднення – як небезпечним факторам для життя (та тим більше, здоров'я людини), що є проблемами оперативного, аварійного чи катастрофічного характеру;

– питанню теплового зараження місцевості (що важливо для аналізу глобальних трансформацій клімату планети), як стратегічному, системному, що визначає майбутній розвиток нашого суспільства в цілому, тому не менш, а навіть більш важливому, ніж питанням протидії поточним аваріям та катастрофам, які завжди носять локальний характер.

Очевидно, що згадані процеси забруднення можуть бути промодельовані на комп'ютері – звичайно, за умови наявності відповідного програмного забезпечення. Результатами такого моделювання для процесів першого класу можуть бути масштаби явища (причому в динаміці, прив'язуючись до дат та, навіть, часу доби) – в першу чергу, питання

територіального поширення забруднення. Такі відомості можуть допомогти спрогнозувати доцільну зону евакуації, зберігаючи людські життя та майно. Економічна обґрунтованість розробки відповідного програмного забезпечення є очевидною (як і його великий соціальний ефект). З точки зору математичної реалізації такої задачі, теоретично можливим є опис усіх трьох типів забруднення, що підпадають під перший клас, однією системою рівнянь (можливо, із зануленням деяких коефіцієнтів у різних випадках із трьох згаданих).

Ще більш важливим з наукової точки зору є моделювання задачі, що була віднесена до другого класу – поширення теплових забруднень, причому у глобальних (або, принаймні, регіональних) масштабах. Результати її розв'язку можна було би використовувати для пошуку критичних областей, що потребують особливої уваги у зв'язку з проблемою глобального потепління.

Зважаючи на вищесказане, можна зробити висновок, що задача моделювання процесу поширення забруднень в цілому є надзвичайно актуальною для сучасної галузі ІТ, а задача моделювання саме теплового забруднення взагалі має стратегічний характер і пов'язана із розвитком усієї людської цивілізації в майбутньому, саме тому вона і буде вирішуватися у даному дослідженні.

1.2 Аналіз науково-технічних джерел з питань моделювання забруднень

Вище було встановлено, що усі процеси забруднення можна укрупнено поділити на:

– такі, що мають причини аварійного (чи катастрофічного) характеру, локальний (максимум регіональний) характер, є тимчасовими (хоча, можливо, із тривалою післядією), наприклад, [1-5];

– стратегічного плану, що зумовлені самою діяльністю людини (зокрема звичайною, повсякденного характеру), мають глобальний характер (впливають на всю планету), постійної (безперервної) дії, наприклад [6-10].

Останній пункт, звичайно, є більш важливим (особливо у довгочасній перспективі), зважаючи на глобальний його характер. Сюди можна віднести:

– теплове забруднення (яке проявляється у явищі глобального потепління);

– біологічне забруднення (зокрема, епідеміологічного характеру, як, наприклад, вірусу герпесу, яким інфіковано за різними оцінками вже більше 90% всього населення планети);

– забруднення океанів твердими побутовими відходами, сміттям, і т.д.;

– різноманітні забруднення хімічного плану (добавки до їжі, активатори, підсолоджувачі, згущувачі, і т.п.), а також, на межі з біологією, – ГМО, тощо.

У всіх вказаних (та більшості інших) випадках «стратегічних» забруднень, самі забруднення можна охарактеризувати певною числовою величиною. Наприклад, для теплового забруднення такою величиною є температура (середня за певний період). Дійсно, саме по зафіксованому значенню температури, спираючись на таблиці середніх температур, характерних для даної місцевості у дану пору року (або за середньорічною температурою), можна судити про ступінь теплового забруднення, причому в динамічному режимі.

1.3 Постановка задачі дослідження

Розглянувши наявні літературні джерела по моделюванню поширення забруднень, можна переходити до формалізації (постановки) задачі дослідження.

У загальному випадку, поточний стан будь-якої системи, зокрема такої, що підлягає забрудненню, характеризується набором з n параметрів (а не однією величиною). Як відомо з вищої математики, упорядкований набір чисел називають вектором, тому можна сказати, що поточному стану об’єкта, наприклад, точки A_i на земній поверхні із заданими координатами (LAT_i; LON_i), відповідає вектор $\vec{x}_i$ значень величин, що описують поточну картину забруднення у цій точці:

$$\vec{x}_i = \{x_{i1}, x_{i2}, ..., x_{in}\}. \qquad (1.1)$$

Сама по собі інформація, яку надає (1.1), уже являє суттєвий інтерес, оскільки дозволяє на основі знання граничних значень контрольованих величин x_{ij} як, наприклад, усілякі ГДК (гранично допустимі концентрації) або температури – для обраного варіанту забруднення, оцінювати ступінь ризику знаходження людини (або тварин, чи рослин, цінних матеріальних об’єктів) у

даній точці A_i. Однак, знання лише одного такого значення вектора $\vec{x}_i$ сильно звужує можливості щодо аналізу часових змін картини забруднення, а знання динаміки процесу у багатьох випадках є більш важливим, ніж наявність інформації про поточний стан об'єкта.

Отже, у найпростішому випадку аналізу теплового забруднення (як уже було сказано вище) можна контролювати тільки один параметр – температуру. При цьому (1.1) перетворюється у найбільш просту свою форму:

$$\vec{x}_i = \{x_{i1}\} = \{x_i\} = \{T_i\}, \tag{1.2}$$

де T_i – поточна температура у точці A_i. В подальшому під вектором (1.1) будемо мати на увазі вираз виду (1.2).

Існує багато способів моделювання стану $\vec{x}_i$: статистичні, кваліметричні, на основі методів сучасної галузі штучного інтелекту, і т.д.

Окремим класом таких систем, які, в принципі, можуть застосовуватися для задач моделювання майбутніх станів забруднення, є рішення на базі нейронних мереж, яким присвячена чи не найбільша відносна частка усіх наукових джерел останніх років з питань прогнозування, наприклад [11-15]. Така ситуація у великій мірі обумовлена певним «бумом», що спостерігається в галузі штучного інтелекту та суміжних напрямках, навколо поняття нейронної мережі. Цей об'єкт широко застосовується для задач розпізнавання, класифікації, апроксимації, а, зважаючи на це – і для прогнозування.

Штучною нейронною мережею (далі – ШНМ), як відомо, називають систему, яка, приймаючи на вхід вектор вхідних величин, за складними (нелінійними) внутрішніми законами виробляє вектор вихідних величин, що є більш-менш задовільним розв'язком поставленої задачі.

Елементарною складовою ШНМ є один нейрон – рис. 1.1. Це об'єкт, що має певну кількість входів (на рисунку – n входів), які називають дендритами та один вихід, що називається аксоном.

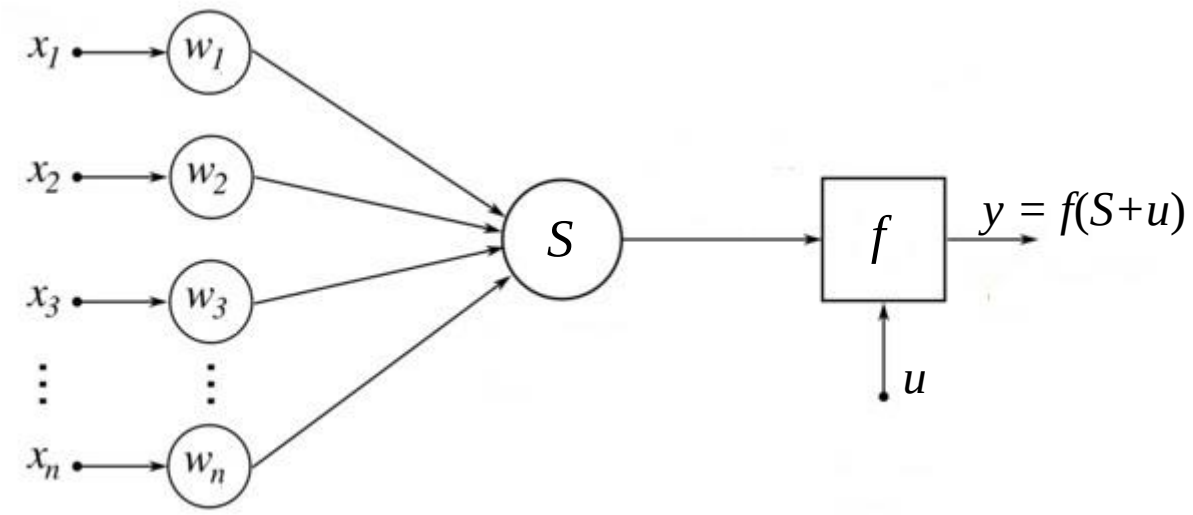

Рисунок 1.1 – Структура штучного нейрону

На входи нейрону подаються вхідні сигнали x_1, x_2, …, x_n, що є виходами попередніх нейронів, розташованих раніше по ходу проходження сигналу. Кожен вхідний сигнал x_i множиться на певне число w_i, що називають вагою (або синаптичною вагою) відповідного входу. Усі таким чином зважені вхідні сигнали подаються на суматор, який виробляє зважену суму S виду:

$$S = \sum_{i=1}^{n} w_i x_i \, . \tag{1.3}$$

До цієї зваженої суми вхідних сигналів може додаватися зсув або зміщення u (у поширеному частинному випадку зміщення відсутнє, тобто $u = 0$) і від цієї суми виробляється певна функція f, що називається активаційною; так і формується вихідний сигнал нейрону y, що передається далі по нейронній мережі:

$$y = f\,(S + u). \tag{1.4}$$

Відповідно комбінуючи (1.3) та (1.4) маємо функцію, що здійснює штучний нейрон:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + u \right). \tag{1.5}$$

При аналізі такого способу завдання функції, що здійснює нейрон, як залежність (1.5), бачимо, що важливу роль для кожного нейрона відіграють три речі (особливо перші дві):

– набір конкретних значень ваг w_i, що описують кожен вхід нейрона;

– вид активаційної функції f даного нейрона;

– наявність зсуву або порогу активації u (який може бути як додатним, так і від'ємним).

Зважаючи на те, що звичайна ШНМ складається як мінімум із кількох нейронів (а частіше – з десятків, чи навіть сотень нейронів), а в кожного з них є багато входів, то задача вибору вагових коефіцієнтів w_i стає дуже серйозною за своєю складністю. Власне, тривалий процес завдання конкретних «правильних» значень цих вагових коефіцієнтів називають процесом навчання нейронної мережі. Існують різні алгоритми навчання нейронних мереж, які укрупнено можна поділити на дві групи: «з учителем» та «без учителя». В цілому, навчання нейронних мереж є настільки складним процесом, що має досліджуватися в окремому великому блоці інформації і не може бути розглянуте в рамках даної магістерської роботи.

Вид активаційної функції f, очевидно, також є дуже важливим моментом для функціонування нейронів, хоча й у значно меншій мірі, ніж вибір синаптичних вагових коефіцієнтів. На рис. 1.2 наведено найбільш поширені види функцій активації, що найчастіше використовуються на практиці при роботі з нейронними мережами. З точки зору обчислювальної складності більш кращим є використання порогових функцій, що фактично являють собою константну залежність (або хоча б лінійних, але не сигмоїдних).

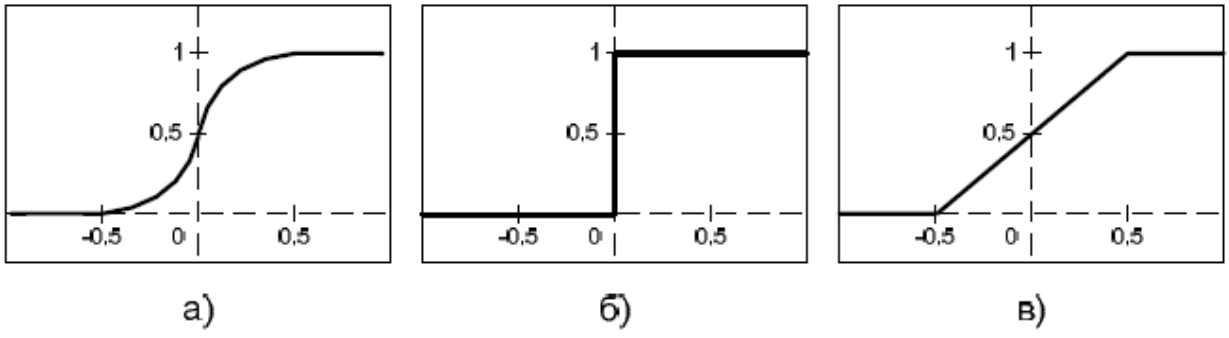

Рисунок 1.2 – Найбільш поширені типи функцій активації: а – сигмоїдна, б – порогова, в – лінійна

При побудові цілої нейронної мережі, крім характеристик окремих нейронів, важливим стає також спосіб їх з'єднання (так само, як і їх загальна кількість, організація нейронів у групи, шари, і т.д.).

Найпростішим і в той же час найпоширенішим способом з'єднання нейронів у конкретну ШНМ є персептрон (перцептрон), що обумовлено широким спектром задач, які можна вирішувати за допомогою ШНМ з такою внутрішньою структурою.

Беручи до уваги усі особливості ШНМ, а також усіх сформованих вище кроків по вирішенню задачі моделювання стану забруднення, бачимо, що з формальної точки зору нейронні мережі є ідеальним інструментом для вирішення задачі прогнозування. В реальності ж ситуація є дещо гіршою: через надзвичайну складність перетворення вхідного сигналу у вихідний, а також через наближений (а не точний) характер результату цього процесу ніхто не може гарантувати, що на досить близьких наборах вхідних параметрів мережі не даватиме кардинально різних відповідей (проблема неперервності та поступової, плавної зміни функції (1.5), яку кінцево і реалізує штучна нейронна мережа у багатовимірному просторі станів системи).

Таким чином, використання нейронної мережі теоретично може мати нестабільний характер і давати погані результати на окремих наборах вхідних даних, навіть при дуже великих обсягах навчальної вибірки та якісно реалізованому процесі навчання. Саме тому часто перевагу надають іншим методам з галузі штучного інтелекту.

Одним із таких популярних методів отримання кінцевого рішення в задачах моделювання складного характеру є застосування апарату нечіткої логіки (fuzzy logic), що є ще одним перспективним механізмом, який також широко застосовується в галузі інтелектуального аналізу даних (ІАД) [16-18]. Перевагою таких систем є можливість роботи із нечисловими, розмитими, або взагалі якісними, вираженими словами, лінгвістичними даними. Процедура фазифікації вхідної інформації, в якій використовуються лінгвістичні змінні дозволяє поставити у відповідність елементам множини $\vec{u}_i$ їх текстові описи, а потім використати простий та зрозумілий набір правил нечітких продукцій для вироблення кінцевого рішення у нечіткій, розмитій формі. Після застосування процедури дефазифікації отримуємо конкретне значення прогнозованої величини. Головною перевагою при такому підході є відсутність необхідності зведення складної математичної моделі (можливо на

основі методів багатовимірного регресійного аналізу, інших статистичних методів), а використання простої бази правил, яку може розробити на основі аналізу логіки роботи системи (або ходу протікання процесу) будь-яка тверезо мисляча людина. Правила мають структуру на зразок «якщо густина населення є високою, то висока імовірність зростання темпів теплового забруднення», «якщо за минулий рік спостерігалася депопуляція населення, то низька імовірність зростання теплового забруднення», і т.п.

Зважаючи на широку поширеність систем нечіткого виводу, розглянемо докладніше весь процес їх використання (зокрема, для задач прогнозування, як у даній роботі). Він полягає у циклічному виконанні процедури нечіткого виводу (див. нижче) за допомогою системи нечіткого виводу. Єдиним кроком, що не виконується під час робочого часу системи, а реалізується лише один раз – при її розробці – є створення бази нечітких правил продукцій (бази правил нечітких продукцій).

Під системою нечітких правил продукцій мається на увазі узгоджена множина окремих правил нечітких продукцій, побудованих на основі виразів виду:

$$\text{ПРАВИЛО <1>: ЯКЩО «}\beta_1 \text{ є } \alpha_{11}\text{» ТО «}\beta_2 \text{ є } \alpha_{21}\text{»,} \qquad (1.6)$$

де «є» означає «являється», або англійською «IS».

В (1.6) нечіткі вирази антецеденту A та консеквенту B розписані у формі «β_i є α_{ij}» через лінгвістичні змінні β_i, деякі з яких є вхідними (це змінні, що беруть участь у антецедентах), деякі – вихідними (змінні, які входять до консеквентів). α_{ij} – який-небудь конкретний терм із терм-множини лінгвістичної змінної β_i.

Таким чином, першим кроком процедури нечіткого виводу, який до того ж виконується лише один раз, є створення бази правил нечітких продукцій. Це задача, яку має виконувати експерт (експерти) у даній предметній галузі, оскільки від якості цієї системи (її повноти та адекватності кожного правила) залежить успішна робота системи прогнозування в майбутньому. Помилки при проектуванні бази правил можуть приводити до катастрофічних наслідків, причому найбільша частка складних проблем має за причину саме недоліки системи нечітких правил продукцій.

Важливою властивістю бази правил має бути її узгодженість, тобто правила не повинні суперечити одне одному (так завжди і є, коли розробку

виконує один експерт, що має закінчений, усталений і ефективний погляд на прогнозування задач заданого класу). При розробці бази правил командою експертів, перевірка на узгодженість має виконуватися окремим підетапом.

Після зведення бази правил нечітких продукцій та її реалізації у певному електронному апаратному забезпеченні розпочинається процес нормального функціонування системи прогнозування, що є ітеративним, циклічним і кожна ітерація складається зі складових, показаних на рис. 1.3:

Рисунок 1.3 – UML-діаграма діяльності процесу нечіткого виводу

– фазифікація вхідних величин, що входять до умов правил нечітких продукцій;

– агрегування підумов для правил, що мають складені антецеденти;

– активізація підвисновків;

– акумулювання висновків (для тих вихідних змінних, різні терми яких входять до різних активних правил нечітких продукцій, а зазвичай для кожної вихідної змінної так і є);

– дефазифікація вихідних змінних.

Розглянемо докладніше кожен із п’яти повторюваних етапів процедури нечіткого виведення.

Першим із них є фазифікація, суть якої полягає у обчисленні значень функцій належності усіх активних термів усіх вхідних лінгвістичних змінних. Даний процес виконується на основі звичайних (не нечітких) даних і має на меті їх переведення у терміни теорії нечітких множин, саме тому фазифікацію ще називають введенням нечіткості.

Ціллю фазифікації є встановлення відповідності між конкретним (зазвичай, числовим) значенням окремої вхідної змінної системи нечіткого виводу та значенням функції належності відповідного їй терма вхідної лінгвістичної змінної. Після завершення цього етапу для всіх вхідних змінних мають бути визначені конкретні значення функцій належності по кожному із лінгвістичних термів, що використовуються у підумовах бази правил системи нечіткого виводу.

Наступним етапом процедури нечіткого виводу є агрегування. Воно застосовується до тих правил, які мають складені умови, що складаються з декількох підумов. Для кожної підумови шляхом виконання процесу фазифікації встановлюється ступінь істинності. По цим даним для кожного правила встановлюється ступінь його істинності в цілому, що виконується на основі об’єднання ступеней істинності окремих підумов, що входять до даного правила (якщо якесь правило побудовано на основі елементарної умови, то ступінь істинності всього правила приймається рівним ступеню істинності цієї єдиної умови). Об’єднання підумов одного правила здійснюється за допомогою операції нечіткої кон’юнкції ТА чи нечіткої диз’юнкції АБО.

Результатом етапу агрегування є встановлення ступеня істинності (у вигляді числа в діапазоні від 0 до 1 включно) для кожного із правил, що утворюють базу нечітких продукцій.

Активізація є третім із повторюваних етапів процедури нечіткого виводу і полягає у встановленні ступеня істинності кожного із підвисновків усіх правил, що були активними на цій ітерації процесу (тобто таких правил, які на попередньому кроці агрегування мали ненульові результуючі ступені істинності). Ціллю активізації є вироблення ступенів істинності для кожного підвисновку, в результаті чого на наступному кроці можна буде перейти до визначення цілих лінгвістичних змінних, що є вихідними.

Акумуляція є передостаннім етапом процедури нечіткого виводу і необхідна через той факт, що різні терми однієї і тієї ж лінгвістичної змінної можуть входити до підвисновків різних правил нечітких продукцій, і тому мають бути якимсь чином об’єднані для формування значення цієї конкретної лінгвістичної змінної.

Результатом акумуляції є сформований набір функцій належності для кожної лінгвістичної змінної, які враховують і відповідають усім термам, задіяним у активованих правилах бази правил нечітких продукцій.

Останній етап процедури нечіткого виводу називається дефазифікацією, а полягає він у тому, щоби на основі розмитої інформації, якою є отримані на попередньому етапі види функцій належності усіх вихідних змінних, обчислити конкретні значення цих змінних, яких їм треба надати.

Для виконання дефазифікації існує багато різноманітних методів, але найбільш популярними є метод центру тяжіння (який також називають просто словом «центроїд») та центру площі, відповідно до яких треба досліджувати геометричну фігуру, обмежену прямими $x = 0$, $x = x_{max}$, $\mu = 0$, $\mu = 1$, а також – лінією, що зображує функцію належності даної вихідної лінгвістичної змінної. У першому випадку слід знаходити координату x центра тяжіння цієї двовимірної фігури, а у другому – координату x вертикальної прямої, що ділить площу цієї фігури на 2 рівних по площі частини.

Після виконання етапу дефазифікації для кожної вихідної змінної буде наявним конкретне числове значення, якому вона повинна дорівнювати для того, щоби забезпечити найкращий результат моделювання.

Нечіткі системи також мають недолік складності контролювання традиційними математичними методами, а результат моделювання сильно залежить від вибору функцій належності (який, взагалі кажучи, є вільним). Через ці причини нечіткі множини у даній роботі застосовувати не будемо.

Як уже зазначалося вище, найбільш повну інформацію про забруднення, зокрема про характер його протікання (воно зменшується, збільшується, чи залишається стабільно постійним), можна отримувати, розглядаючи процес в динаміці, тобто беручи до уваги попередні значення:

$$\vec{x}_{i(1)}, \vec{x}_{i(2)}, ... \vec{x}_{i(m)}, \tag{1.7}$$

де m – загальна кількість значень вектору $\vec{x}_{i(k)}$, що зберігаються з метою аналізу динаміки процесу поширення забруднення (у випадку, що розглядається за основний, замість векторів $\vec{x}_{i(k)}$ можна писати просто середнє значення температури $T_{i(k)}$, див. (1.8) нижче).

Якщо число m є строго фіксованим, то при отриманні нового вектору $\vec{x}_i$, старе його значення має записатися в $\vec{x}_{i(1)}$, а значення $\vec{x}_{i(1)}$ в $\vec{x}_{i(2)}$, і т.д. – усі значення векторів мають зсунутися на одну позицію. Для таких дій зручно використовувати структуру даних «черга» (перший прийшов – перший вийшов, або First Input First Output, FIFO; у термінах, наприклад, стандартної бібліотеки мови C++ це структура типу deque).

Для випадку контролю лише температури, послідовність (1.7) приймає свого найпростішого вигляду:

$$T_{i(1)}, T_{i(2)}, ... T_{i(m)}. \tag{1.8}$$

При збереженні хоча би двох різних значень вектора параметрів забруднення, що описують одну певну точку A_i, необхідною стає процедура порівняння двох таких станів, а точніше, вироблення їх різниці $\Delta\vec{x} = \vec{x}_{i(k)} - \vec{x}_{i(k+1)}$, що для (1.8) є очевидною і рівною простій арифметичній різниці $\Delta T = T_{i(k)} - T_{i(k+1)}$, а у більш загальному випадку (1.7) має розроблятися окремо.

Сукупність значень (1.7) може бути коротко позначена наступним чином:

$$\mathbf{X}_i = \{\vec{x}_{i(1)}, \vec{x}_{i(2)}, ..., \vec{x}_{i(m)}\} = \{\vec{x}_{i(k)} \mid k = 1..m\}, \quad (1.9)$$

де $\mathbf{X}_i$ – матриця, що містить інформацію про усі попередні стани системи, що ураховуються при моделюванні.

Задача оцінки майбутнього стану забруднення $\vec{x}_i$ лише на основі попередніх станів $\vec{x}_{i(k)}$, тобто матриці (1.9), є статистичною задачею побудови часового ряду (зведення тренду), яка може вирішуватися достатньо ефективно, наприклад, шляхом сплайн-апроксимації (при цьому важливу роль грають степінь сплайну, метод здійснення апроксимації і т.д.), а потім – відповідної екстраполяції.

Однак, ефективність моделювання, тобто відповідність результатів прогнозу, який видається математичною моделлю, реальній картині, може бути покращена шляхом урахування другої складової інформації про майбутню картину забруднення, а саме, набору $\vec{u}_i = \{u_{i1}, u_{i2}, ..., u_{il}\}$ наявних значень показників, прямим чином не пов'язаних із $\vec{x}_i$, але, безперечно таких, що на нього впливають. Тут l – кількість таких показників. Моделювання майбутнього стану забруднення на основі переліку наявних показників відповідає кваліметричному підходу (або скоринговому, від score – бали).

Наприклад, для теплового забруднення міської місцевості це може бути густина населення (що спостерігається на даний момент), наявність приросту чи збитку населення (на даний час), і т.п. Ці показники непрямим чином впливають на показник теплового забруднення місцевості у майбутньому.

Відповідно, можна здійснити просту, але ефективну класифікацію методів моделювання забруднень навколишнього середовища, залежно від типу вхідної інформації, що сприймається моделлю. Всього можна виділити три наступних випадки схем моделювання, що розглядаються нижче.

У першій схемі враховується тільки інформація $\mathbf{X}_i$ про попередні стани забруднення, але не враховується наявна інформація $\vec{u}_i$ про вхідні параметри стану, який буде прогнозуватися, тобто:

$$\vec{x}_i = f(\mathbf{X}_i). \quad (1.10)$$

Такий підхід означає вибудовування певного тренду, закону, по якому міняються у часі (мається на увазі дискретний час, коли кожен новий момент

просто відповідає утворенню нового стану системи) параметри стану. Підхід добре працює за відсутності суттєвих несподіваних збурюючих факторів.

Наприклад, у країні з економікою, що активно розвивається, курс національної валюти зростає на кілька десятих долей відсотка щомісяця у порівнянні з курсом долара США. На основі такої інформації вже можна говорити про прогнозне значення курсу валюти через місяць, два, три, але все це дасть більш-менш точний прогноз лише за умови відсутності несподіваних глобальних потрясінь, таких як сильний землетрус, збройна агресія сусідньої держави, банкротство національних компаній чи найбільших корпорацій, і т.п.

Більш витончений приклад використання (1.10) можна навести якраз з галузі моделювання процесу поширення забруднення. Шляхом ретельних спостережень можна встановити, що кожного року середня температура квітня у певній місцевості зростає на 0,1 градуси. Тоді із високим ступенем прогнозованості можна стверджувати, що і у квітні наступного року середня температура буде визначатися цим правилом. При цьому результат отримано лише на основі $\mathbf{X}_i$, без урахування інших, очевидно відомих, обставин $\vec{u}_i$, що спостерігаються у точці моделювання A_i.

Для виявлення трендів використовують методи математичної статистики типу регресійного аналізу, сплайн-апроксимацію, перетворення Фур'є із подальшою фільтрацією, і т.п.

У другій схемі організації моделі враховуються тільки відомі обставини $\vec{u}_i$ майбутнього стану, але без залучення даних про історію попередніх станів $\mathbf{X}_i$:

$$\vec{x}_i = f(\vec{u}_i). \qquad (1.11)$$

У галузі прогнозування така ситуація є досить поширеною, а відповідні моделі називають кваліметричними (скоринговими, бальними, і т.п.). При цьому в (1.11) враховуються поточні значення різноманітних суттєвих показників, важливих для предметної галузі, що розглядається. Ці значення осереднюються, зважуються, згортаються і в результаті, звичайно, отримують якийсь один показник, за величиною якого можна зробити висновок про результат прогнозування.

З математичної точки зору для цього випадку застосовуються різноманітні методи згортки, нормалізації, методи експертного аналізу, і т.д.

Нарешті, у даній роботі пропонується здійснити об’єднання переваг обох підходів (схем) та позбутися їх недоліків шляхом реалізації наступної залежності, де враховуються обидва типи інформації, наявної в системі:

$$\vec{x}_i = f\left(\mathbf{X}_i, \vec{u}_i\right). \qquad (1.12)$$

Такий підхід до моделювання стану забруднення на основі використання вектору $\vec{u}_i$ змінних стану географічної точки, а також з урахуванням картини забруднення у попередні періоди $\mathbf{X}_i$ і буде застосований в рамках програмної реалізації, що виконана у подальшій роботі.

Таким чином, задача дослідження полягає у наступному:

– розробці конкретних методів та особливостей реалізації запропонованого оригінального підходу на базі залежності (1.12) до моделювання процесу розповсюдження теплових забруднень у часі;

– виконанні програмної реалізації розроблених методів та з урахуванням розглянутих особливостей процесу моделювання поширення теплових забруднень у часі;

– аналізі розробленого програмного продукту, що реалізує створену методику, його тестуванні, та виконанні висновків по роботі.

2 ВИБІР ЗАСОБІВ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ МОДЕЛЮВАННЯ ДИНАМІКИ РОЗПОВСЮДЖЕННЯ ЗАБРУДНЕНЬ

2.1 Вибір технології програмування

Першочерговим питання, яке постає перед розробниками практично будь-якого програмного забезпечення, є вибір моделі або технології його розробки. Такими, що реально використовуються на сьогоднішні день у виробничій практиці, є технології структурного (процедурного) та об’єктно-орієнтованого програмування. Кожна з них має свої особливості, переваги і недоліки, які розглянемо докладніше.

Структурне, або як його ще називають практикуючі програмісти, процедурне програмування засноване на використанні окремих структурних блоків - в першу чергу, підпрограм (процедур і функцій).

Історично перші комп’ютерні програми були відносно простими і мали пакетний режим роботи: отримуючи на вхід якусь інформацію (можливо, навіть на перфокарті) вони виконували певний обсяг операцій з обробки цих даних і видавали результат. При цьому існувала сувора функціональна залежність виходу від входу – рис. 2.1.

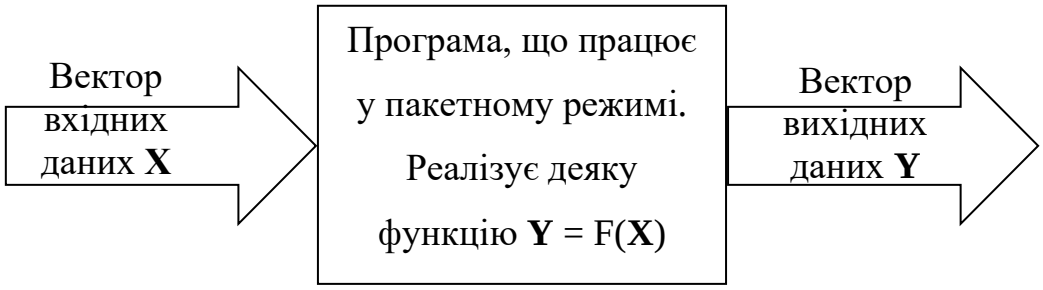

Рисунок 2.1 – Схема роботи пакетної програми

Незважаючи на відсутність явного зв’язку функціональності і структури, пакетні програми в основному мали просту лінійну послідовність виконання. Тобто в них практично відсутні будь-які підпрограми, принаймні, використання підпрограм не було наріжним каменем самої методики програмування.

В міру ускладнення функціональності програмного забезпечення, змінювалася і його внутрішня структура: поступово розвинулася

інтерактивна модель взаємодії користувача і програми – рис. 2.2. Програми стали запитувати інформацію і активно реагувати на дії людини. Ускладнення функціональності привело до відповідного ускладнення програмного коду, якого, в першу чергу, стало просто багато. Багато для того, щоб людина-програміст без всяких спеціальних хитрувань швидко і легко розібралася з незнайомим кодом. Розміщувати код у простій лінійній послідовності без виділення великих блоків стало незручно, в першу чергу, для розуміння цього коду.

Рисунок 2.2 – Схема роботи програми, що активно взаємодіє з користувачем

Тут слід зазначити психологічні особливості сприйняття людиною складних «великих» завдань. Неструктуроване «велике» завдання (наприклад, написання дипломної роботи) зазвичай викликає певний психологічний ступор і, як результат, повну неможливість поступово розібратися з ним. Людині зручно розбити проблему на не надто велику (зазвичай до десятка, а краще 3-4) кількість завдань (наприклад, розділів у дипломній роботі), не замислюючись про реалізацію кожного з них. Коли є ясність і розуміння проблеми на найвищому рівні абстракції, слід приступати до деталізації підзадач, кожну з яких слід розбити на окремі «підпідзадачі» тобто підзадачі нижчого рівня, більш дрібні. Уже після такого розбиття слід аналізувати всі перераховані підзадачі. На певному етапі зупиняються і виконують не розбиття чергової підзадачі на більш дрібні, а безпосередню її реалізацію в програмних кодах.

Отже, можна сказати, що розвиток методики програмування відбувався в ногу з розвитком призначеного для користувача інтерфейсу: і грубо

кажучи, розвиненому інтерфейсу командного рядка відповідає парадигма структурного програмування.

Говорячи більш строго, структурне програмування передбачає побудову програми відповідно до трьох основних принципів: слідування, розгалуження, повторення.

Слідування має на увазі, що оператори і блоки програмного коду слідують і виконуються один за іншим. Розгалуження реалізується різними умовними операторами типу if, і дозволяє вибирати один з декількох подальших варіантів виконання програми. Повторення зазвичай відносять на рахунок циклів (багаторазових повторів однієї і тієї ж ділянки коду, що виконуються підряд), хоча цей же принцип можна віднести і до підпрограм.

Взагалі ж, структурне програмування у деякій мірі є застарілою методикою програмування, на зміну якій разом з віконним інтерфейсом прийшло об’єктно-орієнтоване програмування (деякі сучасні мови програмування загального призначення навіть не дозволяють створити структурну програму, тільки об’єктно-орієнтовану – як, наприклад, Visual C# чи Java). Проте, при створенні невеликих програм (наприклад, до 10000 рядків коду і без передбачуваного розширення) застосування цієї методики програмування більш виправдано і код краще сприймається, ніж його об’єктно-орієнтований варіант.

Суть же методології об’єктно-орієнтованого програмування (ООП) полягає в тому, що система розглядається, як сукупність окремих сутностей – об’єктів, які мають набір якихось своїх внутрішніх параметрів – властивостей, а також можуть взаємодіяти між собою за допомогою деяких дій – викликів методів (або трохи більше непрямим чином – шляхом надсилання повідомлень, оброблюваних методами об’єктів; для цього необхідна присутність активної сутності, яка роздає повідомлення адресатам, як, наприклад, менеджер вікон в ОС Windows).

Якщо говорити про програмний код, то для того, щоб оперувати об’єктом, його спочатку потрібно створити. Об’єкти створюються як змінні, у яких типом виступає клас об’єкта. Клас – це просто опис, які властивості можуть мати об’єкти такого типу (тобто яку інформацію вони можуть зберігати), і які у них є методи (тобто які дії вони можуть виконувати). Об’єкт – це набір значень, чому саме рівні властивості даного об’єкта (свої методи кожен об’єкт отримує від свого класу, тобто методи однакові у всіх об’єктів, що належать даному класу).

Для чого потрібен цей специфічний підхід, адже самі по собі об’єкти не додають нічого корисного (навпаки, введення об’єктів ускладнює програму, вносить в неї нові сутності)? Виявляється, реалізовуючи всі сутності, необхідні, згідно з алгоритмом, для роботи програми, у вигляді класів і об’єктів, ми спрощуємо її розуміння для самих себе. Саме тому ОО-підхід рекомендується до застосування для великих проектів (більше десятків тисяч рядків коду), коли утримувати «в голові» всю систему цілком стає важко. Можна сказати, що розбиття програми на об’єкти і проектування їх класів наближає розуміння предметної області до звичного людського образу мислення (в разі «великих» проектів). Людина мислить класами, об’єктами і зв’язками між ними.

Порівняльна складність проектів, що мають однакову функціональність, але побудованих по-різному (згідно структурному і об’єктно-орієнтованого підходів до програмування), як функція їх обсягу, може бути побудована за даними [19] і показана на рис. 2.3. З графіка рис. 2.3 слід, що, якщо потрібно реалізувати продукт з невеликою функціональністю (тобто кількість рядків коду, що її реалізують буде очевидно невеликою, порядку кількох тисяч рядків), то це краще робити без застосування класів, так як вони будуть тільки ускладнювати всю справу. Якщо ж програма має більш-менш значну функціональність, а значить, реалізується хоча б кількома тисячами рядків коду, то вже є сенс замислюватися про застосування об’єктно-орієнтованого підходу. Однозначно будуватися за принципами ООП повинні програми, які мають 10000 рядків коду і більш.

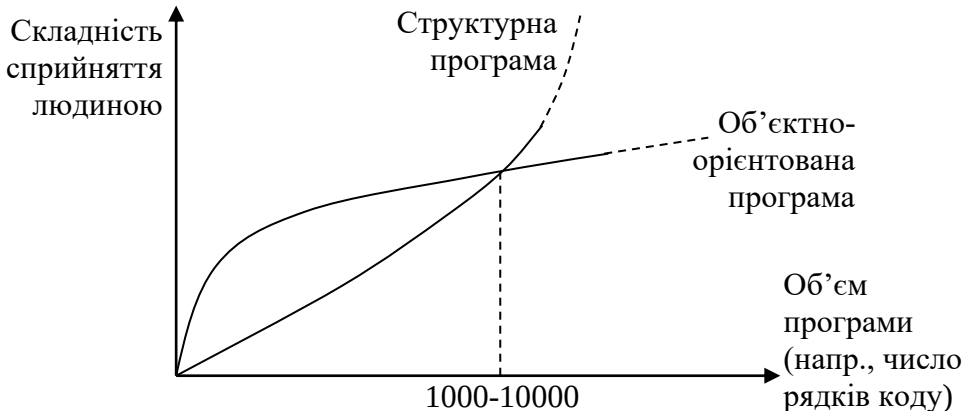

Рисунок 2.3 – Порівняльна складність висхідного тексту двох програм, що мають однакову функціональність, але реалізованих по-різному: згідно об’єктно-орієнтованому та структурному підходам

Відзначимо, що часто крім розглянутих міркувань, також на вибір методики програмування впливають інші чинники, наприклад, можливість майбутнього розширення функціональності, створення якомога більш зрозумілого коду (для роботи над проектом цілої команди, а не одного програміста), або просто побажання замовника застосувати найбільш сучасний підхід до програмування.

Крім розбиття (декомпозиції) всієї предметної області на об'єкти (класи) і співвідношення між ними, також ОО-підхід має на увазі дотримання трьох основних його принципів: інкапсуляція, наслідування, поліморфізм.

Під інкапсуляцією мається на увазі об'єднання даних (значення властивостей класу у деякого конкретного об'єкта) та засобів їх обробки (методи класу). Це знову ж таки зручно психологічно, так як дозволяє реалізовувати окремі завершені сутності – класи, які самі обробляють свої дані. Звернення до об'єктів цих класів відбувається за допомогою методів, що утворюють інтерфейс класу.

Наслідування дуже корисно, тому що дозволяє сильно скоротити обсяги повторюваного коду (до чого потрібно завжди прагнути при розробці будь-якого програмного забезпечення). Згідно з цим принципом виділяється клас, який має загальний набір властивостей і методів для декількох більш розширених класів. Цей клас оголошується батьком, базовим класом для декількох похідних від нього (нащадків, спадкоємців). Всі класи-нащадки успадковують від базового всі його властивості та методи, але до цього ще мають свої власні оригінальні властивості і / або методи.

Наприклад, клас Студент є похідним від класу Людина, тому що кожна людина має властивість Ім'я, Прізвище, метод Відпочити(). Однак у Студента є свої специфічні властивості і / або методи, які як раз і відрізняють його від просто Людини: СереднійБал, НомерЗаліковки, ЗдатиЕкзамен(), і т.д.

При наслідуванні іноді методи батьківського і похідного класу мають однакове призначення, але реалізовуються по-різному. Такі методи називаються перевантаженими. Наприклад, метод Відпочити() у класу Людина реалізується як відпочинок на дивані, а у класу Студент – як похід в клуб. При цьому ще раз підкреслимо, що призначення методу в обох випадках одне і той саме.

Поліморфізм є можливістю деякої функції приймати об'єкти як батьківського, так і похідних класів, і вміти викликати перевантажені методи

саме того класу, об’єкт якого був переданий в функцію. Слід сказати, що це досить специфічна можливість і в загальному багато програмістів використовують ОО-підхід і без звернення до поліморфізму.

Нехай, наприклад, у програмі є функція ПровестиВихідні(), припустимо яка не належить якомусь класу (хоча це не принципово). Нехай аргументом цієї функції є об’єкт класу Людина. Тоді в неї можна передавати об’єкти всіх похідних від Людини класів: Студент, Службовець, Пенсіонер, і т.д., тому що всі вони є Людиною (спадкоємці цього класу). Ясно, що ця функція повинна включати різні дії: ПрибратиКвартиру(), ПітиНаРинок(), і в тому числі Відпочити(). Так ось поліморфізм дозволяє всередині цієї функції просто вказати назву методу Відпочити(), не вказуючи якого саме класу він повинен бути викликаний, а вже в процесі виконання програми, якщо в функцію переданий об’єкт класу Студент, то викликається саме його метод Відпочити(), а якщо переданий об’єкт класу Пенсіонер, то автоматично викликається саме його метод Відпочити(), і т.д. Кажуть, що функція ПровестиВихідні() – поліморфна, і вона є такою завдяки тому, що реалізує принцип поліморфізму.

Важливими поняттями в ООП також є: статичні члени класу, абстрактні методи і класи, дружба функцій і класів, і т.д.

Ґрунтуючись на перерахованих особливостях двох існуючих методів програмування, вибираємо структурний підхід як більш простий, що відповідає невеликим (не промисловим) масштабам проектованого ПЗ, а також вимогам до складності (програма не повинна бути складною через цільову аудиторію користувачів, якими є не спеціалісти з галузі ІТ, а кліматологи).

2.2 Обґрунтування вибору мови програмування

Після вибору технології програмування наступним кроком є вибір мови програмування (або комплексу мов – за умови, наприклад, написання програмного продукту у вигляді розподіленого клієнт-серверного додатку, а також при деяких інших умовах).

Найбільш укрупнено за типом платформи кінцевого програмного продукту усі мови програмування можна розподілити на засоби для розробки настільних додатків, засоби веб-розробки та мобільної розробки. Мобільні технології в даній роботі взагалі не можуть стати в нагоді, оскільки

розроблюваний програмний продукт не потребує мобільності, а обчислювальні ресурси мобільних систем в рази менше, ніж у настільних, чи тим більше, клієнт-серверних. Що ж стосується альтернативи «веб-додаток» чи «настільне програмне забезпечення», перший варіант видається зручнішим через наступні причини:

– є повністю кросплатформенним (якісні веб-додатки однаково функціонально працюють у будь-яких сучасних веб-браузерах, незалежно від того, на якій операційній системі вони запущені);

– дозволяє використовувати обчислювальні ресурси серверу, які завжди значно потужніші, ніж у настільних ПК;

– можуть мати досить швидкий та зручний інтерфейс, за умови використання у них сучасних веб-технологій (типу AJAX, кешування, локальної доробки даних, предзавантаження сторінок, і т.д.).

Таким чином, обираємо веб-розробку та проаналізуємо засоби, що можуть при такому випадку бути використані.

Сучасні засоби для розробки Інтернет-ресурсів є настільки обширними, що розібрати їх усі в рамках дипломної роботи абсолютно не уявляється можливим. Тому в процесі вибору будемо орієнтуватися як на об’єктивні фактори (такі, як, наприклад, підтримка сучасних технологій програмування), так і суб’єктивні, але такі, що набули широкого впливу (як-то схильність окремих людей до певних мов, чи, навіть, стилів програмування, що у результаті спричинює популярність відповідних засобів розробки на масовому рівні).

Таким чином, розглянемо найпоширеніші засоби розробки, що є популярними на даний момент у відповідній галузі.

В цілому увесь процес веб-розробки традиційно ділять на дві компоненти: front-end та back-end – рис. 2.4.

2.2.1 Засоби Front-end розробки

Спрощено кажучи, під «передньою» частиною – Front-end – у програмуванні мають на увазі те, що бачить користувач. Для настільних систем під цим мають на увазі інтерфейс, а під back-end’ом мають на увазі логіку роботи програми, яка теоретично (зокрема, так часто роблять у Linux-системах) взагалі може бути реалізована окремою програмою із текстовим консольним інтерфейсом. Задачею фронт-енду при цьому є зручний для широкого кола користувачів збір усіх вхідних даних, необхідних для

виконання змістовних операцій консольною утилітою back-end'ом. У багатьох же випадках фронт та бек поєднані разом в рамках одного програмного забезпечення. Такою є ситуація для настільних систем.

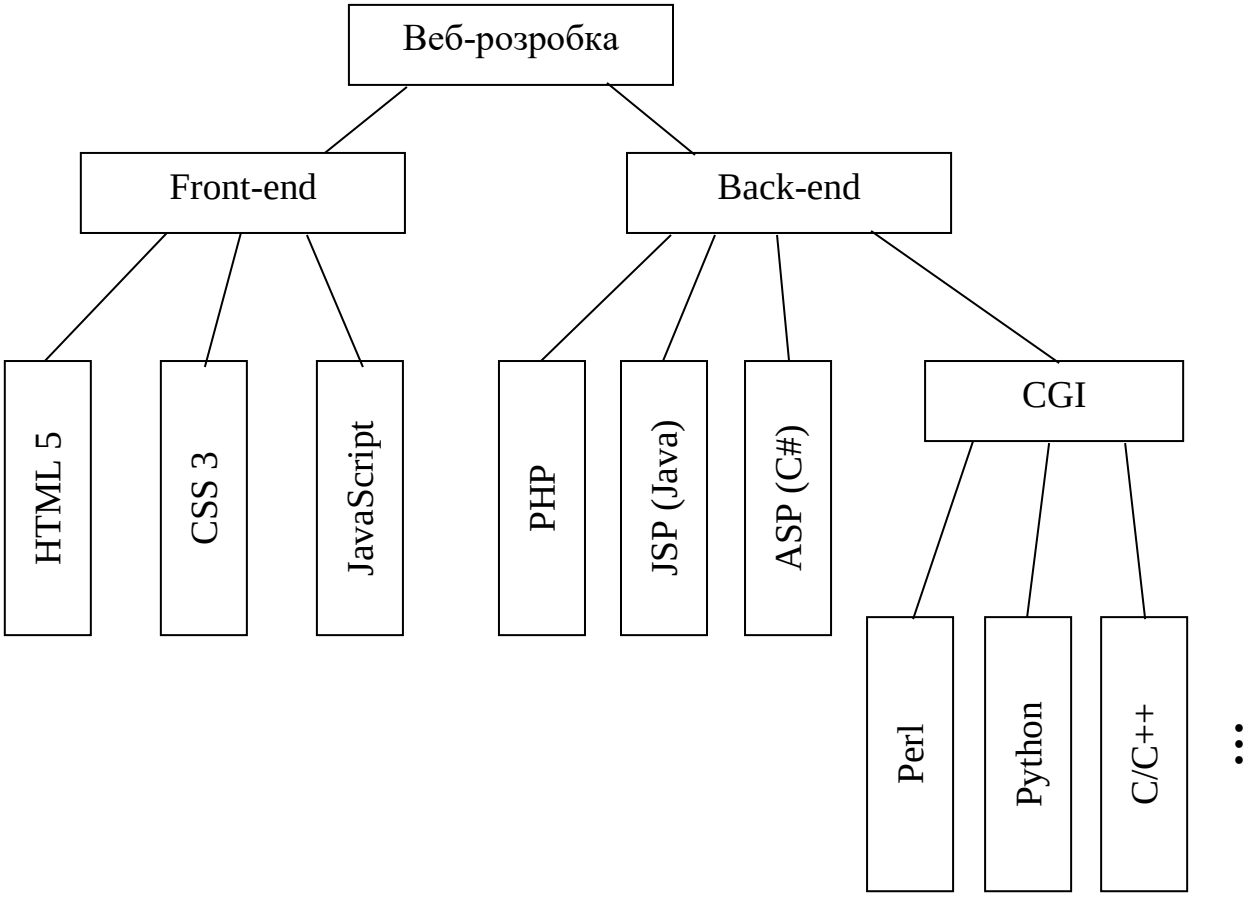

Рисунок 2.4 – Найпопулярніші засоби розробки сучасних Інтернет-ресурсів

Якщо конкретніше говорити про веб-програмування, то тут ситуація є максимально поляризованою (ще більше, ніж у описаному вище випадку написання фронт-енду одним програмістом для бек-енд-утиліти, створеної іншим розробником), оскільки користувач працює і «бачить» систему через браузер на одній віддаленій машині (на комп'ютері, що називається клієнтом), а логіка додатку та практично уся супутня інформація розміщується на зовсім іншому комп'ютері-сервері.

Таким чином, усі програмні коди та висхідні тексти, що виконуються у браузері, тобто на стороні клієнта, відносяться до front-end частини. З іншого боку, все, що виконується на сервері, відноситься до back-end складової.

Якщо глибоко не вдаватися у детальний аналіз, то коротко можна сказати, що набір інструментів для front-end розробки значно менший, ніж для беку. Серед них можна виділити:

– HTML – п’ятої версії (HTML5), найсучасніший стандарт мови гіпертекстової розмітки (HyperText Markup Language), де традиційно під гіпертекстом мають на увазі текст, оснащений гіперпосиланнями на інші частини документів та, можливо, супроводжуваний картинками, звуками, відео, і т.п.;

– CSS – третьої версії (CSS3) – правила запису стилів різноманітних елементів веб-сторінки, або, якщо говорити точно – каскадні таблиці стилів;

– JavaScript – найпотужніший інструмент front-end розробки, що дозволяє вивести html-сторінки на новий рівень практично повноцінних додатків, на зразок настільних, що активно реагують на дії користувача, передають і отримують інформацію, від нього.

Ці технології є безальтернативними, тому розглянемо їх докладніше.

Hyper Text Markup Language (HTML) – мова розмітки гіпертексту – призначена для написання усіх документів в мережі World Wide Web – WWW.

HTML документ – це текстовий файл, який має спеціальні мітки, що називаються тегами, які при затребуванні клієнтом передаються із комп’ютера-серверу на браузер клієнта і використовуються ним для відображення вмісту файлу па екрані комп'ютера.

За допомогою цих міток можна виділяти заголовки документа, змінювати колір, розмір і написання літер, вставляти графічні зображення і таблиці. Але основною перевагою гіпертексту перед звичайним текстом є можливість додавання до вмісту документа гіперпосилань – спеціальних конструкцій мови HTML, які дозволяють клацанням миші перейти до перегляду іншого документа. Таким чином «гіпертекст» означає «надтекст» – розвинення ідеї текстового документу, але із більш широкими можливостями.

Сама по собі мова HTML є різновидом більш загальної мови XML (eXtensible Markup Language), причому з одного боку звужує її властивості, а з іншого – навпаки деталізує. Так, у мові XML допустимим є використання будь-яких тегів, тобто з довільними іменами, в той час, як назви тегів мови HTML є цілком фіксованими, а інші слова окрім стандартних, використовувати не можна. З іншого боку, той вміст веб-сторінки, що розміщений біля конкретних тегів (точніше – між парою однакових тегів), набуває визначених стандартом властивостей (на відміну від тексту, що розміщений поруч із тегами XML, які самі по собі практично нічого не

означають, а використовуються лише при наявності певних домовленостей про відображення й обробку того, чи іншого тегу) відображується браузером відповідно до суті цих тегів: якщо тегом є <B>, то текст зображуватиметься жирним, <i> – курсивом, і т.д.

HTML-документ має дві складові: власне текстова інформація, тобто дані, що складають вміст документа, і теги – спеціальні конструкції мови HTML, які використовуються для розмітки документа і керують його відображенням. Теги мови HTML визначають, в якому вигляді буде представлений текст, які його компоненти будуть виконувати роль гіпертекстових посилань, які графічні або мультимедійні об'єкти повинні бути включені в документ.

Графічна та звукова інформація, що включається в HTML-документ, зберігається в окремих файлах. Програми перегляду HTML-документів називають браузерами, і вони інтерпретують теги розмітки і оперують текстом і графікою, розміщуючи їх на екрані відповідним чином. Для файлів, що містять HTML-документи використовується розширення .htm або .html.

У переважній більшості випадків теги використовуються парами. Пара складається з тега, що відкриває <ім'я_тега> і тега, що закриває </ім'я_тега>. Дія будь-якого парного тега починається з того місця, де зустрівся відкриваючий тег, і закінчується при зустрічі відповідного закриваючого тега. Часто пару, що складається з відкриваючого і закриваючого тегів, називають контейнером, а частину тексту між відкриваючим і закриваючим тегом – елементом.

Послідовність символів, яка утворює текст, може складатися з пробілів, табуляцій, символів переходу на новий рядок, символів повернення каретки, букв кириличного та латинського написань, знаків пунктуації, цифр, і спеціальних символів (наприклад #, +, $, @), за винятком наступних чотирьох символів, що мають в HTML спеціальний сенс: < (менше), > (більше), & (амперсанд) і «(лапки)».

Найпершим із тегів HTML розміщується однойменний тег <html>. Він завжди відкриває документ, так само, як тег </html> повинен неодмінно стояти в останньому його рядку. Ці теги позначають, що рядки, які між них знаходяться, представляють собою єдиний гіпертекстовий документ. Без цих тегів браузер або інша програма перегляду не в змозі ідентифікувати формат документа і правильно його інтерпретувати.

Підсумовуючи можливості мови HTML, можна сказати, що за допомогою різних тегів можна реалізовувати таблиці, форматувати текст, вставляти в документ зображення, відео-, звукові файли та інше. В процесі свого розвитку все більша увага приділялася тегу <style> та пов’язаними з ним каскадними таблицями стилів.

Каскадні таблиці стилів (Cascading Style Sheets, CSS) – це мова, яка містить набір властивостей для визначення зовнішнього вигляду документа. Специфікація CSS (CSS3 – на даний момент) визначає властивості і описову мову для встановлення зв'язку властивостей з елементами в документі. Розуміння таблиць стилів необхідно для додавання динамічного стилю сторінки. Під динамічним стилем (dynamic style) тут маємо на увазі модифікацією таблиці стилів, пов'язану з документом за допомогою сценарію.

На вузлі консорціуму W3C (www.w3.org) можна знайти останню інформацію про нововведення і елементах, підтримуваних таблицями стилів.

Таблиці стилів представляють собою абстракцію, в якій стиль документа визначається окремо від змісту або структури. Існує три методи додавання таблиць стилів в документ, доступних для Web-майстра – в цілому, з підвищенням рівня складності розширюються надані можливості з одночасним збільшенням ступеня абстракції. Перший метод полягає в використанні таблиці внутрішніх стилів (inline style sheet). Внутрішні стилі (inline styles) визначаються безпосередньо в елементі. Другий метод полягає в використанні таблиці глобальних стилів (global style sheet) для визначення стилю на початку документа. Третій, найбільш абстрактний і потужний метод полягає в використанні таблиці пов'язаних стилів (linked style sheet) для визначення стилю окремо в іншому документі.

Внутрішні стилі мало відрізняються від традиційного HTML. При використанні внутрішніх стилів зовнішній вигляд документа важко змінити. Перевага даного методу полягає в скороченому обсязі розмітки і в тому, що HTML може бути більш повноцінно використаний для подання вмісту презентації. Використання таблиці глобальних стилів дозволяє більш ефективно відокремити представлення від вмісту, а також дає можливість швидкої і незалежної зміни стилю і обробки документа. Використання таблиці пов'язаних стилів дає більше переваг, представляючи вміст у вигляді набору сторінок або визначаючи цілий Web-вузол за допомогою одного файлу.

Термін cascading (каскадні) в назві CSS вказує на можливість злиття різних таблиць стилів для створення єдиного визначення стилю для елемента або для цілого документа. Це дозволяє проводити передбачуване злиття таблиці стилів Web-вузла з таблицею стилів документа і навіть з внутрішнім стилем.

Отже, внутрішній стиль (inline style) є по суті таблицею стилів для одиночного екземпляру елемента і його визначено безпосередньо у тегу елемента. Таблиця внутрішніх стилів визначається з використанням атрибута STYLE, а дані для атрибута визначаються за допомогою мови таблиці стилів.

Внутрішні стилі допомагають при вивченні мови CSS або за необхідності швидко змінити одиночний екземпляр елемента. Однак, внутрішні стилі не відповідають ідеології структурованого документа і погано працюють при необхідності зміни зовнішнього вигляду ряду елементів в документі, коли презентація та вміст розділені в повному обсязі. Для відділення стилю документа від його структури таблиця стилів повинна бути визначена в заголовку документа або як окремий файл, який пов'язаний з документом.

Наступною, більш ідеологічно вірною можливістю, є використання окремого тегу <STYLE> для завдання таблиць глобальних стилів, який зазвичай розміщується в заголовку документа. Розміщення усіх стилів документа в одному місці спрощує зміну режиму відтворення документа. Наведений нижче приклад таблиці стилів визначає зовнішній вигляд усіх параграфів в документі. Для зміни режиму відтворення параграфів потрібно змінити тільки елемент STYLE. Якби використовувалися внутрішні стилі, то довелося б міняти кожен параграф в документі окремо.

Для зв'язку стилю з певним елементом використовується селектор (selector). У наведеному вище прикладі був створений простий селектор, який пов'язаний зі стилем у всіх параграфах. Можуть бути також визначені більш функціональні контекстуальні селектори, що описуються нижче у цьому розділі.

Ще одним із трьох способів завдання стилів є введення таблиці пов'язаних стилів (linked style sheet), яка знаходиться у зовнішньому файлі. Перевага використання таблиці пов'язаних стилів полягає в тому, що всі правила і стилі можуть бути визначені і вміщені в одному файлі, який може бути спільно використаний багатьма сторінками або навіть цілим Web-вузлом. При використанні таблиці пов'язаних стилів обробка всіх параграфів

цілого Web-вузла може бути змінена в одному документі. Таблиця пов'язаних стилів може також підвищити продуктивність, оскільки вона кешується локально на диску клієнта, окремо від документа, так що кожен документ має менший розмір, а інформацію про стилі буде потрібно завантажити тільки один раз.

Для визначення таблиці пов'язаних стилів у заголовку документа розміщується тег <LINK>. Його атрибут REL визначає, що пов'язаний файл є таблицею стилів, а атрибут TYPE визначає тип MIME таблиці стилів. Атрибут HREF є покажчиком URL, що вказує на зовнішню таблицю стилів. Таблиця пов'язаних стилів повинна містити тільки контекстуальні правила і визначення стилю і не може включати код HTML.

Для створення таблиці стилів всередині документа використовується той же синтаксис, який використовувався при створенні таблиці пов'язаних стилів. Мова CSS складається з селекторів і правил подання (presentation rules). Селектори (selectors) визначають елементи, які пов'язані з певним правилом, а правила подання встановлюють методи обробки даних елементів.

CSS містить два типи селекторів: прості і контекстуальні. Простий селектор пов'язує елемент на основі його атрибутів або типу незалежно від контекстуального положення усередині інших елементів. Контекстуальні елементи є більш потужними – вони можуть зв'язати правило з контейнерами певного елемента, наприклад, усі теги <EM> всередині тегів <P>.

У базовій формі простий селектор може бути створений для зв'язку певного елемента, класу елементів або ідентифікатора (ID) з певним стилем. Селектори можуть бути перераховані через кому, що дозволяє одночасно описати кілька селекторів. Контекстуальні селектори визначають ієрархію контейнерів, з якою повинен бути пов'язаний стиль. Ієрархія контейнерів визначається порядком елементів в списку через кому. Кожен селектор може посилатися на теги CLASS, ID або тип елемента. Всі елементи контекстуального селектора є нечутливими до регістру – наприклад, .cool це те ж саме, що і .cOoL.

Псевдоклас (pseudo-class) складається з елементів одного типу, які задовольняють певному контекстуальному критерію. Наприклад, переглянуті елементи Anchor (посилання) представляють собою псевдоклас visited. Активні і не переглянуті посилання являють собою псевдокласи active і link, відповідно. Псевдоклас в таблиці стилів відділяється двокрапкою.

Псевдоклас може бути використаний так само, як клас або покажчик ID і є нечутливим до регістру.

На одні й ті ж самі елементи можуть посилатися кілька селекторів. CSS визначає послідовність каскадування (cascading order), яка використовується для вирішення проблем перекривання областей дії селекторів і правил. Послідовність каскадування об'єднує всі правила, які застосовуються до елементу, шляхом сортування на основі їх визначень. Наприклад, елемент Strong, що знаходиться в елементі H1, може мати правила подання, визначені селектором H1, селектором STRONG і контекстуальним селектором для елементів Strong всередині елементів H1. Параметр каскадування в CSS визначає порядок об'єднання даних трьох правил. Загалом, правило для більш конкретного контекстуального селектора відкидає правило для менш конкретного селектора, а правила, подані нижче у початковій таблиці стилів або документі, мають більш високий пріоритет.

Докладно розглянувши каскадні таблиці стилів слід також подивитися основні особливості останнього, дуже потужного засобу front-end-розробки, а саме – мови програмування JavaScript. Це спеціалізована мова програмування, яка традиційно використовується для управління об’єктною моделлю документа у браузері під час перегляду сторінок мережі Інтернет (існують продукти, що перетворюють JavaScript на мову загального призначення, команди якої виконуються на сервері, наприклад, Node.JS; але такий підхід сильно протирічить початковій концепції використання цієї мови програмування, і, за умови наявності значної кількості традиційних засобів back-end-розробки перетворення на серверну мову ще й JavaScript значна кількість програмістів вважає абсолютно недоцільною, тому про цей варіант використання говорити більше не будемо).

Особливістю JavaScript є те, що його висхідні програмні коди інтерпретуються, що є досить гнучким, але повільним рішенням.

Оператори у цій, в цілому C-подібній мові, розділяються крапкою з комою. Мова чутлива до регістру, що часто випускають з виду початківці, ймовірно тому, що HTML, застосовуваний зазвичай з JavaScript спільно, не залежить від регістру (імена тегів і атрибутів HTML можна писати як малими, так і великими літерами).

Однорядковий коментар виділяється символом //, а багаторядковий - парою символів / * і * /, в чому JavaScript знову повторює C.

До даних застосовується слабкий (динамічний) контроль типів. В операторах з різнотипними даними останні автоматично приводяться до необхідного типу. Типи даних можуть бути примітивними і складеними. Примітивні типи містять прості однорідні значення, такі дані можна передавати функціям як параметри за значенням, а не за посиланням. Складені типи містять різнорідні дані (в тому числі і складені), їх передають у функції тільки за посиланням.

Мова JavaScript об’єктно-орієнтована, проте заснована на прототипах, а не на класах. Є чотири типи об’єктів: вбудовані об’єкти, об’єкти браузера, об’єкти документа і об’єкти користувача (програміста).

Введення-виведення в основному обмежене взаємодією з документами і користувачами. За умовчанням передбачається, що доступ до локальної файлової системи заборонений. Однак браузери можуть надавати спеціальні об’єкти, за допомогою яких забезпечується робота з файловою системою користувача, хоча і з видачею попереджень про небезпеку виконання файлових операцій.

Сценарії JavaScript активно взаємодіють з об’єктами, вбудованими в Web-сторінку. Для цього вони, власне, і створюються. Але перш, ніж ця взаємодія стане можливою, слід впровадити код сценарію в текст HTML-документа. Існує кілька способів зв'язати HTML-документ з конкретним сценарієм (скриптом), але зазвичай їх просто розміщують всередині контейнерного тега <SCRIPT>, тобто між дескрипторами <script> і </script>.

Контейнер <SCRIPT> в цьому випадку буде перебувати безпосередньо в HTML-документі, причому у довільному його місці. Програмний код пишуть прямо в HTML-документі або в спеціальних текстових файлах, які можна викликати з головного HTML-документа. Для початку розглянемо перший варіант. Перш за все браузер знаходить тег <script> в тілі веб-документа, і весь наступний текст намагається обробити як скриптовий код. І так до тих пір, поки не зустріне закриваючий тег </script>. Після цього всі наступні символи будуть вважатися HTML-текстом. Будь-який HTML-документ може містити довільне число «скриптових включень», але кожне має відкриватися і завершуватися відповідним тегом. Від їх розташування у тілі HTML-документа іноді може залежати функціонування всієї Web-сторінки, але про це буде сказано пізніше.

Контейнерний тег <script> може містити атрибут SRC, який вказує ім’я або URL-адресу текстового файлу, що містить код сценарію. Цей атрибут

необхідний в тому випадку, якщо сценарій розташований не безпосередньо в HTML-документі, а в окремому файлі. Розширення файлу зі сценарієм може бути яким завгодно, але зазвичай використовують js.

Якщо сценарій розташовується в окремому файлі, то в ньому, зрозуміло, теги <SCRIPT> і </ SCRIPT> не пишуть. Сценарій, завантажений з зовнішнього файлу, можна уявити собі просто як його вставку в HTML-документ.

У браузерах, що потенційно підтримують сценарії, цю функцію користувач може відключити (зокрема, із міркувань підвищеної безпеки). Крім того, існують браузери, які принципово не підтримують сценарії. У даній ситуації бажано хоча б вивести повідомлення про те, що на сторінці був сценарій, але в даному конкретному випадку він не виконується. З цією метою в HTML-документі використовують контейнерний тег <noscript>, в якому розміщують текст, який з'явиться користувачеві за умови, що сценарій з тієї, чи іншої причини не виконуватиметься. Всі браузери, які підтримують сценарії, проігнорують вміст тегів <noscript> крім тих випадків, коли підтримка сценаріїв відключена. Браузери, що принципово не підтримують сценарії, навпаки, вміст тега <script> опустять (особливо, якщо він вкладений у теги <!-- -->, які є HTML-коментарем), а тега <noscript> – відобразять.

Як уже зазначалося, в окремих файлах зазвичай розміщують бібліотеки функцій (визначення функцій), а також сценарії, які використовуються в декількох HTML-документах одного або декількох сайтів. Сценарій можна також писати у вигляді рядка операторів, між якими ставиться крапка з комою. Такий рядок, взятий в лапки, служить у якості значення атрибута-події.

Виклики функцій і їх визначення можуть слідувати в довільному порядку, але тільки якщо вони розташовані в одному і тому ж контейнері <script>. Якщо вони розміщуються у різних контейнерах <script>, то необхідно, щоб визначення функції передувало її виклику. Аналогічно, якщо визначення функцій знаходяться в окремому файлі, то його необхідно завантажити в HTML-документ раніше викликів цих функцій.

При спробі завантажити і виконати неправильний варіант HTML-коду з'явиться діалогове вікно з повідомленням «Помилка: Передбачається наявність об'єкта». Браузер інтерпретує теги HTML послідовно. Так, зустрівши вираз виклику функції myfunc() в одному контейнері <script>, браузер ще не має в пам'яті визначення цього об'єкта (функції),

розташованого в іншому контейнері <script>. Якщо ж визначення функції і її виклик знаходяться в одному і тому ж контейнері <script>, то його вміст спочатку завантажується в пам'ять, а потім аналізується. Коли інтерпретатор зустрічає виклик функції, то він шукає її визначення в пам'яті і в разі успіху виконує код цієї функції.

Якщо необхідно, щоб сценарій виявився в браузері перш, ніж буде завантажено елементи HTML-документа, то його слід розташувати у верхній частині HTML-коду, а ще краще в контейнері <head> в заголовку документа.

2.2.2 Засоби Back-end розробки

Як зазначалося вище, під Back-end'ом мають на увазі програмні компоненти, що працюють на сервері, і вибір засобів для реалізації цих компонентів є вкрай широким – рис. 2.4, справа.

Для цілей даного дослідження будемо використовувати просте підключення до бази даних із отриманням довідкової інформації про температури у заданому місті, яке обиратиме користувач. Для таких, порівняно простих задач, доцільно використати найпоширенішу на сьогоднішній день мову серверного програмування PHP, яка найчастіше всього використовується із СУБД MySQL. Переважна більшість сайтів і веб-сервісів в Інтернеті написано за допомогою PHP. За деякими оцінками PHP застосовується більш ніж на 80 % сайтів, серед яких такі сервіси, як facebook.com, vk.com, baidu.com і інші. І така популярність не видається дивною. Простота мови дозволяє швидко і легко створювати сайти і портали різної складності.

PHP був створений в 1994 році датським програмістом Расмусом Лердорфом і спочатку являв собою набір скриптів на іншій мові програмування Perl. Пізніше цей набір скриптів був переписаний в інтерпретатор на мові C. Із самого виникнення PHP представляв зручний набір інструментів для спрощеного створення веб-сайтів і веб-додатків.

Розглянемо, які ж переваги надає PHP:

– для всіх найбільш поширених операційних систем (Windows, MacOS, Linux) є свої версії пакетів розробки на PHP, а це означає, що можна створювати веб-сайти для будь-якої з цих операційних систем;

– PHP може працювати в зв'язці з різними веб-серверами: Apache, Nginx, IIS, тощо;

– простота і легкість освоєння є особливістю цієї мови. Як правило, маючи навіть невеликий досвід в програмуванні на PHP, можна створювати простенькі веб-сайти;

– PHP схожий на мову C, тому, знаючи C, або одну з мов з C-подібним синтаксисом, буде простіше опановувати PHP;

– PHP підтримує роботу з великою кількістю систем баз даних (MySQL, MS SQL Server, Oracle, Postgres, MongoDB, та інші);

– поширеність хостингових послуг і їх дешевизна. Так як, як правило, хостингові компанії розміщують веб-сайти на PHP на веб-серверах Apache або Nginx, які працюють на одній з операційних систем сімейства Linux. І веб-сервери, і операційні системи на базі Linux безкоштовні, що знижує загальну вартість використання хостингу;

– постійний розвиток, адже PHP продовжує розвиватися, виходять все новіші версії, які несуть нові функції, адаптуючи мову програмування до нових викликів. І, як правило, перехід на нову версію не викликає ніяких труднощів;

– існує надзвичайно багато якісних Інтернет-ресурсів з підтримки процесу програмування на PHP, де можна задати власне питання і найскоріше отримати кваліфіковану відповідь.

Зважаючи на усі позитивні риси, мова PHP і була обрана у якості основного засобу Back-end розробки у даному проекті.

Ще одним важливим елементом Back-end складових є система управління базами даних, вибір яких на сьогоднішній день є надзвичайно широким.

В першу чергу, оберемо бази даних реляційного типу (коли уся інформація зберігається у вигляді пов’язаних між собою таблиць, доступ до яких здійснюється за допомогою мови SQL), оскільки відомості, що зберігаються у базі даних є простими цифровими (значення температур у різних місцевостях). Такі дані носять довідковий характер, використовуються в подальшому для аналітичного математичного аналізу (статистичного), і в цілому не передбачають глибокої обробки засобами штучного інтелекту. Відповідно, використання традиційних реляційних баз даних для такого випадку є оптимальним варіантом.

Серед реляційних баз даних на сьогоднішній день досить широко використовуються miniSQL, MySQL, PostgreSQL та Oracle. Перша СУБД є дуже обмеженою, тому зустрічається досить рідко серед реальних хостинг-

провайдерів. Останній варіант є найпотужнішим і має настільки широкі можливості по обробці даних, що для даного випадку вони є надмірними. Крім того, Oracle є пропрієтарною системою із досить коштовними ліцензіями. Такий продукт для цілей даного дослідження в цілому немає ніякого сенсу застосовувати.

Реальну конкуренцію на ринку для «середніх» додатків можуть становити MySQL та PostgreSQL, але для цілей даної роботи цілком вистачить можливостей MySQL. Про її достатні можливості по обробці даних свідчить той факт, що уже багато років вона є найбільш поширеною в Інтернет СУБД, яка є доступною абсолютно у будь-якого більш-менш серйозного провайдеру. PostgreSQL має деякі додаткові можливості, що проявляються у дещо якіснішій обробці корпоративних даних великих обсягів, але для цілей даної роботи ця особливість не є перевагою, тому у якості СУБД обрано MySQL (або, як її більш правильно називати, MareaDB).

Таким чином, у якості мов програмування для обраного напрямку (веб-розробки) обираємо HTML, CSS, JavaScript та PHP, СУБД MySQL.

2.3 Вибір середовища розробки

Останнім питанням є вибір конкретного середовища розробки, що дозволяє писати код на цих обраних мовах. У якості такого середовища можна взяти популярний на сьогоднішній день серед студентської спільноти Visual Studio Code, або більш професійний PHP Storm, або простий текстовий редактор типу Sublime Editor. Така широта вибору засобів обумовлена простотою архітектури програмного рішення, що вирішує поставлену задачу (зокрема, пакетним режимом роботи, і т.п.) та стандартністю (поширеністю) обраних мов програмування серед представників сучасної програмістської спільноти.

У якості середовища розробки оберемо популярне завдяки своїй простоті та зручності (але достатньо функціональне) середовище Visual Studio Code, що має цікаві можливості.

Це редактор вихідних кодів, розроблений Microsoft для Windows, Linux і MacOS. Позиціонується як «легкий» редактор коду для кросплатформенної розробки веб- і хмарних додатків. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторингу. Має широкі можливості для кастомізації: призначені для

користувача теми, поєднання клавіш і файли конфігурації. Розповсюджується безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом, але готові збірки розповсюджуються під пропрієтарною ліцензією.

Visual Studio Code заснований на Electron-фреймворку, що дозволяє з використанням Node.js розробляти настільні додатки, які працюють на движку Blink. Незважаючи на те, що редактор заснований на Electron, він не використовує редактор Atom. Замість нього реалізується веб-редактор Monaco, розроблений для Visual Studio Online.

Однією із найзручніших функцій цього середовища є згадана вище IntelliSense. Це технологія автодоповнення Microsoft, найбільш відома в Microsoft Visual Studio, яка дописує назву функції та інші ключові слова при введенні початкових літер. Крім прямого призначення, IntelliSense використовується для доступу до документації та для усунення неоднозначності в іменах змінних, функцій і методів, використовуючи рефлексію.

Як і інші системи автодоповнення, IntelliSense є зручним способом переглянути описи функцій, в тому числі переліки їхніх аргументів. Вона прискорює розробку ПО, зменшуючи кількість імен і параметрів, які програміст повинен тримати в пам'яті. Крім того, вона зменшує кількість необхідних запитів до документації, виводячи частину документації у вигляді спливаючих вікон в редакторі коду. В ході роботи IntelliSense формує в пам'яті базу даних, що містить метадані класів, змінних і інших конструкцій, які використовуються в додатку, що розробляється. «Класична» реалізація IntelliSense працює, знаходячи в коді спеціальні маркери, такі як символ точки. Як тільки користувач вводить один з таких маркерів після імені сутності, яка містить один або кілька доступних членів (таких як змінні або методи), IntelliSense показує користувачеві спливаюче вікно зі списком відповідних членів.

Середовище Visual Studio Code має ще багато дуже зручних можливостей, через які воно, власне, і набуло своєї популярності. Це середовище взято у якості основного засобу розробки у даній роботі.

Таким чином, у даному розділі здійснили докладне обґрунтування засобів, які доцільно застосувати для подальшої реалізації системи моделювання динаміки поширення забруднень (теплового), що буде виконано у наступному розділі.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ МОДЕЛЮВАННЯ ДИНАМІКИ РОЗПОВСЮДЖЕННЯ ЗАБРУДНЕНЬ

3.1 Проектування інтерфейсу користувача системи

При створенні програмного продукту наступним кроком після вибору технології програмування, мови програмування та основного засобу розробки є проектування інтерфейсу користувача системи, тобто її зовнішнього вигляду. Він формується набором та розміщенням необхідних елементів управління, які призначені для збору вхідної інформації від користувача та для відображення йому вихідної інформації по роботі програмного продукту.

Беручи до уваги специфіку проектованого програмного продукту (який має надати температурний прогноз на майбутній період на основі набору статичних вхідних даних), можна зробити висновок, що у нього буде 2 основних екрани: для збору інформації та для відображення результатів прогнозування. Таким чином, якщо класифікувати інтерфейс користувача, то, незважаючи на складність самої задачі, що вирішується, інтерфейс користувача хоча і є графічним, але досить простим і може бути віднесений до пакетних – рис. 2.1.

Ця теза напряму витікає із алгоритму роботи користувача із програмним забезпеченням, що розробляється, і який зображено на рис. 3.1.

На першому екрані, що наведено на рис. 3.2, при старті розміщено потрібну кількість елементів управління для введення необхідної інформації. А саме, у першому горизонтальному ряді розміщена інформація про середні температурні показники даної місцевості за останні p років (за умовчанням $p = 10$, а в цілому дане значення p відображується в окремому полі для редагування, де його можна змінювати).

Далі наводиться перелік показників, які є важливими для процесу виділення теплоти (у регіональному масштабі) і напроти кожного виділено поле для введення поточного значення цього показника. Поруч із кожним показником розміщується його вага (ваговий коефіцієнт w), що встановлюється експертним шляхом.

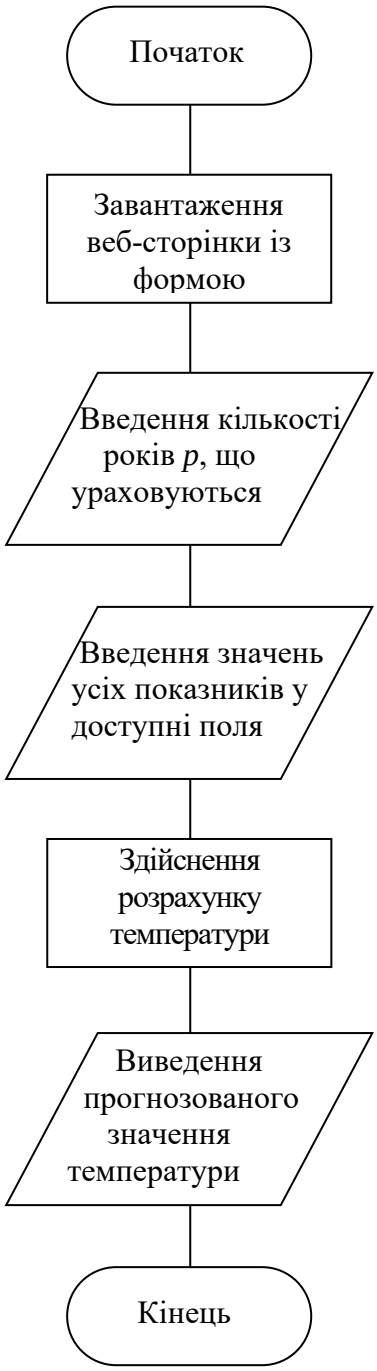

Рисунок 3.1 – Алгоритм роботи користувача із проектованим ПЗ

Виділено наступні показники для оцінки теплового забруднення у містах:

– загальна чисельність населення;

– густина населення (якщо та ж сама кількість енергії виділяється на меншій площі, забруднення при цьому є більшим);

– відсоток приросту загальної чисельності населення за минулий рік (деякі міста розвиваються повільно, деякі швидко, а інші взагалі занепадають);

– середня зарплатня (багатше населення споживає більше енергії);

– оцінка ступеня насиченості міста промисловими підприємствами (наприклад, за 5 бальною шкалою – від 0 до 5 включно);

– наявність теплових або атомних електростанцій у 30-км зоні поруч із містом (найбільші генератори тепла).

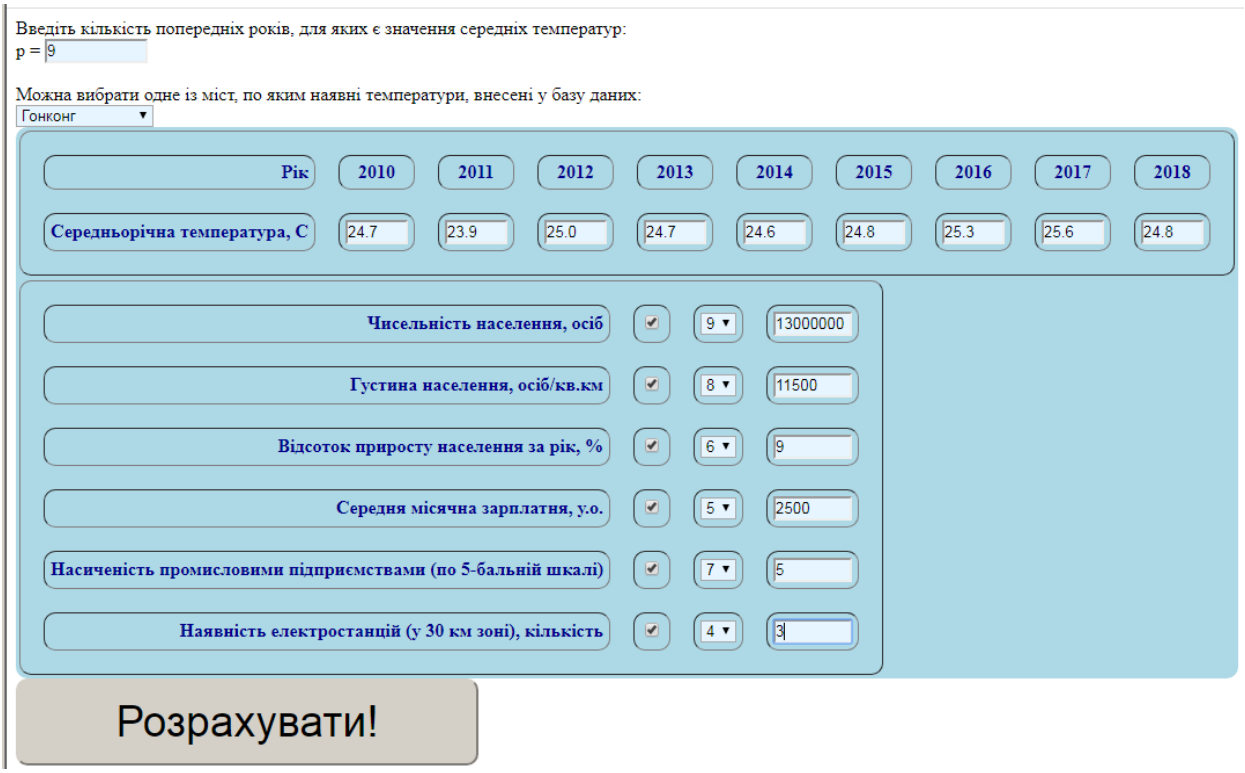

Рисунок 3.2 – Зовнішній вигляд екрану розробленого програмного забезпечення для введення інформації

Рисунок 3.3 – Екран програмного забезпечення для виведення інформації

Зовнішній вигляд вікна браузеру із сторінкою для введення вхідних даних показаний на рис. 3.2. Після введення усіх необхідних даних користувач повинен ініціювати процес розрахунку, для чого використовується спеціальна кнопка «Розрахувати». Після натиснення на неї відбувається розрахунок за формулами, що описані у наступному підрозділі, а результат видається користувачеві у вигляді простого скалярного значення середньорічної температури, яка, як прогнозується, буде спостерігатися у заданій точці A_i наступного року – показано на рис. 3.3. Відмітимо, що графік тут генерується динамічно, відповідно до кількості років спостереження.

3.2 Опис застосованих у роботі алгоритмів

Як уже зазначалося вище, для урахування різноманітних за своєю природою, і, навіть, структурою, показників в кваліметрії їх піддають наступним операціям:

– нормалізації (зазвичай, усі показники приводять до безрозмірного вигляду, наприклад, діленням на максимальне значення, яке взагалі може спостерігатися для кожного показника, що розглядається);

– зважуванню (зазвичай простим множенням вагового коефіцієнта на значення показника – якщо він підсилюючий, стимулюючий, або множенням вагового коефіцієнта на обернену до показника величину – якщо він понижуючий, дестимулюючий);

– згортці (об’єднання у єдиний загальний результат усіх окремих нормалізованих і зважених показників).

Ці операції можуть виконуватися багатьма способами, серед яких найбільш часто вживаються наступні три комбінації, які уособлюються формулами:

– мультиплікативна формула:

$$x_i = \frac{\prod_{k=1}^{l_1} \left(u_{ik} \cdot \rho_{ik} \cdot w_{ik} \right)}{\prod_{k=l_1+1}^{l_1+l_2} \left(u_{ik} \cdot \rho_{ik} \cdot w_{ik}^{-1} \right)},$$

де l_1 – кількість вхідних показників, при збільшенні яких підвищується значення x_i, такі показники, відповідно до термінології [9], будемо називати стимулюючими, підсилюючими (показники-стимулятори);

ρ_{ik} – коефіцієнт для приведення до єдиної міри (розмірності) k-того показника, нормалізуючий множник, який часто розраховується як $1/u_{ik\,\max}$.

При такому підході також слід розглядати константу l_2 – кількість вхідних показників, при зменшенні яких підвищується прогнозоване значення x_i (їх будемо називати дестимулюючими, показники-дестимулятори, понижуючі). Справедливим є співвідношення:

$$l_1 + l_2 = l;$$

– формула на базі теорії адитивної корисності:

$$x_i = \frac{\sum_{k=1}^{l_1}\left(u_{ik} \cdot \rho_{ik} \cdot w_{ik}\right)}{\sum_{k=l_1+1}^{l_1+l_2}\left(u_{ik} \cdot \rho_{ik} \cdot w_{ik}^{-1}\right)};$$

– формула згідно теорії адаптивної корисності:

$$x_i = \sum_{k=1}^{l_1}\left(u_{ik} \cdot \rho_{ik} \cdot w_{ik}\right) + \sum_{k=l_1+1}^{l_1+l_2}\left(\frac{w_{ik}}{u_{ik} \cdot \rho_{ik}}\right). \qquad (3.1)$$

Останній варіант і узято в [9] та ряді схожих робіт (з невеликими варіаціями) за основу. Конкретніше, методика моделювання з [9] складається з трьох кроків:

– обчислення відносних значень кожного показника (тобто приведення до єдиної міри), іншими словами це аналог обчислення добутків $u_{ik} \cdot \rho_{ik}$;

– обчислення загального показника зростання x_i шляхом множення кожного нормалізованого показника на його ваговий коефіцієнт та сумування відповідно до формули (3.1);

– прогнозування нової температури шляхом множення наявного її значення на поправочний коефіцієнт, який також нормалізується, відповідно до формули (3.2):

$$T_i = T_{i-1} \cdot \left(1 + \frac{x_i}{x_m}\right), \qquad (3.2)$$

де x_m – максимальне значення поправочного коефіцієнту, яке досягається за умови, що усі добутки $u_{ik} \cdot \rho_{ik}$ досягають своїх екстремальних значень і рівне простій сумі усіх вагових коефіцієнтів w_{ik}.

Таким є підхід, оснований на урахуванні поточних значень показників, які, на думку експертів, є важливими для зміни температури в майбутньому по відношенню до наявного її значення.

Розглянемо також існуючий підхід до побудови часового тренду, що ефективно використовується в такому розділі математичної статистики, як регресійний аналіз. У випадку, що розглядається, маємо певну зміну температури у часі. Говорячи строго математичною мовою одна величина *T* (температура) залежить від іншої величини *t* (часу). Оскільки зараз буде розглядатися відомий блок матеріалу із галузі регресійного аналізу, то, як прийнято у цій галузі, для аргументу будемо тимчасово використовувати позначення *x*, а для функції – *y*. Після встановлення фінального виду формули, що буде використовуватися в даній роботі, повернемося до позначень *t, T*.

Отже, задачі відтворення такої функціональної залежності між однією величиною (аргументом) та іншою (функцією) є типовими для розділу математичної статистики, що називається регресійним аналізом. Сам вид цієї функціональної залежності називається регресією (або рівнянням регресії). Іноді назву уточнюють словом «одновимірна», якщо треба підкреслити, що вихідна величина залежить тільки від однієї вхідної.

В загальному випадку предметом пошуку у регресійному аналізі може бути не тільки функція однієї змінної. З математичного аналізу добре відомі функції багатьох змінних, які також широко використовуються у регресійному аналізі, якщо необхідно встановити залежність однієї вихідної величини від декількох вхідних аргументів. Така задача називається множинною регресією (або встановленням рівняння множинної регресії). Розглянемо ці задачі докладніше.

Як зазначено вище, у практичних дослідженнях часто виникає необхідність встановлення залежності між однією вихідною величиною та

набором вхідних параметрів, які впливають на значення цієї величини. Результатом є функція багатьох змінних:

$$y = f(x_1, x_2, \dots, x_n).$$

У найпростішому і найбільш поширеному випадку розглядається залежність вихідної змінної від одного аргументу:

$$y = f(x). \qquad (3.3)$$

Функція (3.3) може бути більш чи менш складною, залежно від явища, що розглядається.

Наприклад, експериментально досліджується залежність середньої кількості опадів (у мм), що випадають у певній місцевості, від кількості днів, що пройшли від початку сезону дощів. На основі реальних опадів побудовано графік, на якому кожна точка відповідає одному дощу: по осі абсцис відкладається номер дня від початку сезону дощів, на який спостерігався дощ, а по осі ординат – кількість мм води, що випала під час цього дощу – рис. 3.4. Шляхом регресійного аналізу можна приблизно побудувати функціональну залежність, за допомогою якої для інших днів можна спрогнозувати кількість опадів, які в середньому випадатимуть у даній місцевості.

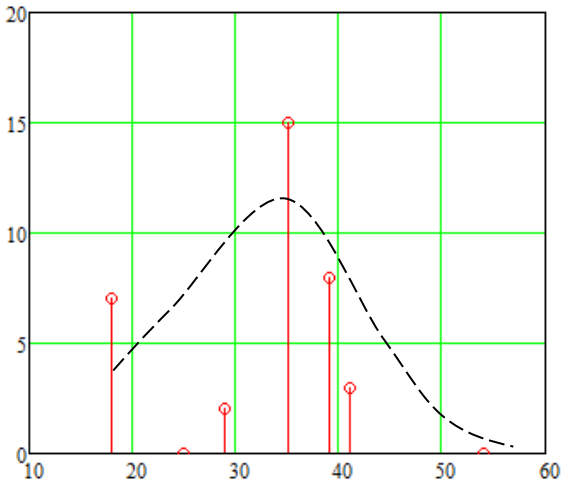

Рисунок 3.4 – Приклад побудови регресії для залежності середньої кількості опадів (у мм), залежно від часу (у днях), що пройшов від початку сезону дощів (нелінійна регресія)

Очевидно, залежність рис.3.4 є досить складною з математичної точки зору. У найпростішому випадку функція (3.3) представляється лінійною залежністю:

$$y = ax + b, \qquad (3.4)$$

де коефіцієнти a і b мають бути визначені на основі експериментальних даних про процес, що моделюється.

Наприклад, виконано декілька експериментальних замірів величини y_i при заданій величині x_i – рис. 3.5, що показані точками.

.

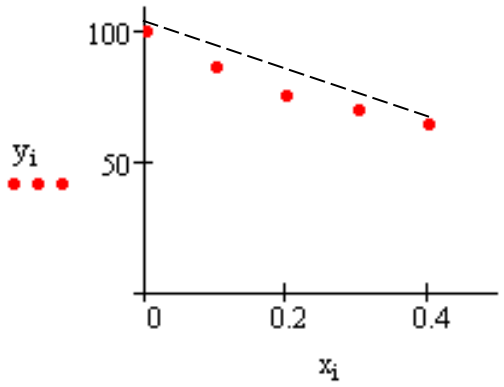

Рисунок 3.5 – Функціональна залежність, близька до лінійної

Після виконання експериментів постає задача відтворення функціональної залежності величини y від x. Як уже сказано вище, можливо виконати апроксимацію лінійною залежністю (3.4) (і яка показана пунктиром на рис. 3.5). Можливо, більш адекватно відображувала би явище квадратична залежність (або які-небудь інші варіанти функціональної залежності), але з математичної точки зору, як найпростіша, узята саме лінійна залежність (3.4).

Обрати конкретні коефіцієнти можна за методом найменших квадратів: слід розглянути відхилення $\varepsilon_i = (y_i - (ax_i + b))$ та знайти мінімум суми квадратів таких нев'язок:

$$\Sigma\varepsilon_i^2 \rightarrow \min.$$

Із цієї вимоги слідує система двох рівнянь, з якої можна визначити коефіцієнти лінійної регресії:

$$\begin{cases} a\sum x_i^2 + b\sum x_i = \sum x_i y_i \\ a\sum x_i + bn = \sum y_i \end{cases} \rightarrow \quad a = \frac{\frac{\sum x_i y_i}{n} - x_{cp} y_{cp}}{\frac{\sum x_i^2}{n} - \left(x_{cp}\right)^2}; \qquad b = y_{cp} - a x_{cp}. \quad (3.5)$$

Аналогічну до (3.5) систему можна отримати і для коефіцієнтів квадратичної регресії:

$$\begin{cases} a\sum x_i^4 + b\sum x_i^3 + c\sum x_i^2 = \sum x_i^2 y_i \\ a\sum x_i^3 + b\sum x_i^2 + c\sum x_i = \sum x_i y_i \\ a\sum x_i^2 + b\sum x_i + cn = \sum y_i. \end{cases}$$

Невідомі коефіцієнти a, b, c можна знайти, розв'язуючи отриману систему лінійних алгебраїчних рівнянь, наприклад, методом Крамера (звичайно, можливим є застосування і інших підходів):

$$a = \frac{\Delta_a}{\Delta} = \frac{\begin{vmatrix} \sum x_i^2 y_i & \sum x_i^3 & \sum x_i^2 \\ \sum x_i y_i & \sum x_i^2 & \sum x_i \\ \sum y_i & \sum x_i & n \end{vmatrix}}{\begin{vmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_i \\ \sum x_i^2 & \sum x_i & n \end{vmatrix}},$$

$$b = \frac{\Delta_b}{\Delta} = \frac{\begin{vmatrix} \sum x_i^4 & \sum x_i^2 y_i & \sum x_i^2 \\ \sum x_i^3 & \sum x_i y_i & \sum x_i \\ \sum x_i^2 & \sum y_i & n \end{vmatrix}}{\begin{vmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_i \\ \sum x_i^2 & \sum x_i & n \end{vmatrix}},$$

$$c = \frac{\Delta_c}{\Delta} = \frac{\begin{vmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 y_i \\ \sum x_i^3 & \sum x_i^2 & \sum x_i y_i \\ \sum x_i^2 & \sum x_i & \sum y_i \end{vmatrix}}{\begin{vmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_i \\ \sum x_i^2 & \sum x_i & n \end{vmatrix}}.$$

Зважаючи на складність (непрактичність) формул квадратичної регресії, в роботі будемо застосовувати лінійну регресію, розмірність якої встановимо нижче.

Переходячи до питання розмірності, можна згадати, що на практиці одна контрольована величина може бути функцією багатьох вхідних параметрів:

$$y = f(x_1, x_2, \ldots, x_m).$$

Форма вказаної залежності може бути досить складною (залежно від конкретного процесу, що досліджується), але з математичної точки зору найпростішим є відтворення рівняння лінійної множинної регресії:

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_mx_m. \qquad (3.6)$$

Задача багатовимірного регресійного аналізу полягає у визначенні коефіцієнтів a_i на основі доступних експериментальних даних. Під експериментальними даними мається на увазі таблиця відповідності виміряних значень вхідних параметрів x_{ij} та значення контрольованої величини y_i, загальний вид якої показано як табл. 3.1.

Таблиця 3.1 - Загальний вид таблиці експериментальних висхідних даних для зведення рівняння множинної регресії.

№ експ	x_1	x_2	…	x_m	y
1	x_{11}	x_{12}	…	x_{1m}	y_1
2	x_{21}	x_{22}	…	x_{2m}	y_2
…	…	…	…	…	…
n	x_{n1}	x_{n2}	…	x_{nm}	y_n

Для виконання конкретних математичних операцій працюють з трьома матрицями, що описуються нижче.

Першою записується регресійна матриця:

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1m} \\ 1 & x_{21} & x_{22} & \ldots & x_{2m} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{nm} \end{pmatrix}.$$

Далі береться матриця-стовпець значень контрольованої величини:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}.$$

Останньою формується матриця-стовпець шуканих коефіцієнтів:

$$Y = \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_m \end{pmatrix}.$$

Використовуючи метод найменших квадратів, операції диференціювання за компонентами вектора, та виконуючи необхідні перетворення, отримують наступну формулу для знаходження невідомих коефіцієнтів рівняння (3.6) множинної лінійної регресії:

$$A = (X^T X)^{-1} X^T Y, \qquad (3.7)$$

де літерою T позначена операція транспонування (заміна рядків на стовпці, або навпаки);

степенем мінус 1 позначена обернена матриця.

Як бачимо, залежність (3.7) є дуже складною, оскільки вимагає перемноження матриць, а головне, знаходження оберненої матриці. Для цих операцій необхідним є використання спеціалізованого математичного програмного забезпечення. Дійсно, немає жодного сенсу використовувати складні алгоритмічні прийоми на мові програмування загального призначення (як, наприклад, PHP або JavaScript), якщо існують спеціалізовані продукти, в яких ці операції реалізуються однією командою (такі, як, наприклад, Mathcad, і т.п.).

Таким чином, за умови використання мови загального призначення, реалізовувати доцільно лише одновимірну регресійну модель, тому у якості основи для написання програмного забезпечення приймемо залежність (3.4), коефіцієнти якої шукають за формулами (3.5), які проаналізуємо докладніше.

У (3.5) наявні деякі стандартні величини, що підлягають визначенню:

– середнє арифметичне (або мовою математичної статистики – математичне очікування) вхідної величини x, яка є аргументом функції (3.4);

– середнє арифметичне вихідної величини y, що є функцією у залежності (3.4);

– середнє квадратичне вхідної величини x (сума квадратів усіх спостережених x, поділена на їх загальну кількість);

– середній добуток величин x та y.

Усі ці величини можна розраховувати в одному циклі від 0 до n (не включаючи), по закінченні якого можна виконати арифметичний розрахунок отриманих середніх значень і визначити числа a та b відповідно до (3.5).

Користуючись блок-схемами алгоритмів розрахунку коефіцієнтів лінійної регресії, що наведені на рис. 3.7 – рис. 3.8, можна розпочинати процес реалізації програмної системи, проекту програмного забезпечення, що призначене для моделювання процесу теплового забруднення місцевості у часі.

Окрім самої математичної процедури обчислення коефіцієнтів лінійної регресійної моделі (3.4), також необхідні деякі прості (технічного характеру) речі, пов’язані із введенням-виведенням даних. Зважаючи на статистичний характер роботи даного програмного забезпечення, йому необхідні певні вхідні дані, що мають невеликий обсяг і являють собою значення середніх температур, що спостерігалися у даній місцевості протягом останніх p років. Зважаючи на те, що кількість років p не може бути дуже великою, доцільно організувати просте ручне введення цих температур у відповідні текстові поля, що і реалізовано раніше – рис. 3.2, рядок у верхній частині вікна програми. Також у програмі реалізовано завантаження із бази даних значень температур останніх років для деяких раніше введених у базу міст. Тоді робити для них прогноз можна на основі зміни значень показників, досліджуючи, як зміняться передбачувані значення, якщо змінити умови, в яких знаходиться на даний момент та, чи інша місцевість.

Таким чином, розглянуто відомий із математичної статистики підхід до зведення регресійної моделі на прикладі лінійної регресії.

Нарешті можна розглянути суть запропонованого нового підходу до моделювання майбутніх станів процесу поширення теплового забруднення, яка полягає в об’єднанні формул (3.1)-(3.2) із моделлю (3.4)-(3.5). Відповідна формула наводиться нижче, а отримана вона шляхом підстановки (3.5) до (3.4), а потім використанням результату у (3.2), куди також підставлена (3.1).

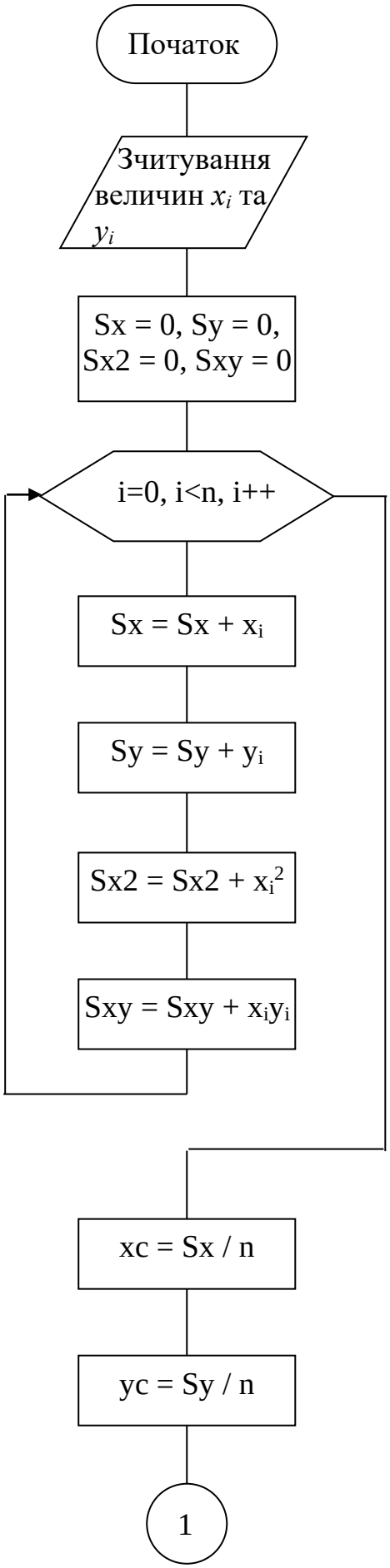

Рисунок 3.7 – Алгоритм розрахунку лінійної регресії (початок)

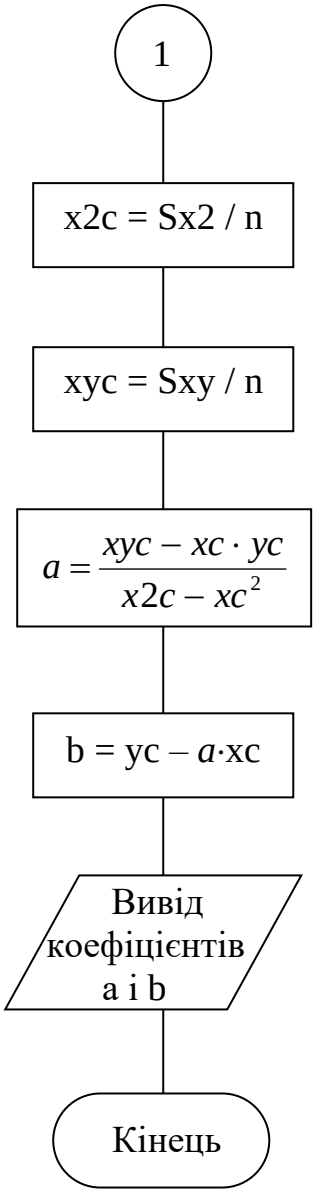

Рисунок 3.8 – Алгоритм розрахунку лінійної регресії (продовження)

$$T_i = \left(\frac{\frac{\sum t_i T_i}{n} - t_{cp} T_{cp}}{\frac{\sum t_i^2}{n} - \left(t_{cp}\right)^2} t + \left(T_{cp} - \frac{\frac{\sum t_i T_i}{n} - t_{cp} T_{cp}}{\frac{\sum t_i^2}{n} - \left(t_{cp}\right)^2} t_{cp} \right) \right) \times$$

$$\times \left(1 + \frac{\sum_{k=1}^{l_1} \left(u_{ik} \cdot \rho_{ik} \cdot w_{ik} \right) + \sum_{k=l_1+1}^{l_1+l_2} \left(\frac{w_{ik}}{u_{ik} \cdot \rho_{ik}} \right)}{x_m} \right).$$

Виконуючи прості арифметичні перетворення та підстановки, отримаємо кінцеву формулу для розрахунку майбутньої температури:

$$T_i = \left(\frac{\frac{\sum t_i T_i}{n} - t_{cp} T_{cp}}{\frac{\sum t_i^2}{n} - \left(t_{cp}\right)^2} \left(t - t_{cp}\right) + T_{cp} \right) \cdot \left(1 + \frac{\sum_{k=1}^{l_1} \left(u_{ik} \cdot \rho_{ik} \cdot w_{ik}\right) + \sum_{k=l_1+1}^{l_1+l_2} \left(\frac{w_{ik}}{u_{ik} \cdot \rho_{ik}} \right)}{\sum_{k=1}^{l_1+l_2} w_{ik}} \right). \quad (3.8)$$

Залежність (3.8) являє собою основну математичну залежність (математичну модель) по якій досліджується процес поширення теплового забруднення місцевості (а саме, прогнозування значення температури майбутнього періоду), і яка реалізована програмно.

3.3 Особливості програмної реалізації застосованих у роботі алгоритмів

Програмний продукт, що розробляється, має чимало досить складних особливостей, серед яких, найголовніше, виділяється власне розрахункова процедура, яка виконується за формулою (3.8). Вона, як і багато інших функціональних речей, реалізована функцією мови JavaScript. Наведемо перелік функцій, що використовуються у програмному продукті разом із їхніми короткими описами:

– filldiv() – заповнює основний шар (maindiv), вибудовуючи в ньому таблицю для введення даних потрібної форми та розміру. Ця функція активно використовує число nump, яке вводиться у верхній частині сторінки і за умовчанням дорівнює 10. При зміні цього числа вся таблиця для введення даних динамічно оновлюється. Так само відбувається при виборі одного із стандартних міст, температури яких наявні у базі даних (тоді кількість таких температур для даного обраного міста і є новим значенням числа nump;

– startcalc() – реалізує формулу (3.8) і побудована на базі алгоритмів, розглянутих у попередньому підрозділі;

– checknum() – сервісна функція, яка перевіряє, щоби користувач вводив у поля тільки числові дані, а у протилежному випадку очищує введене (нечислове) значення і видає відповідне попередження. Ця ж функція обробляє і зміну поля для числа nump, перебудовуючи в цьому випадку весь головний шар;

– filltemperatures(e) – це функція обробник AJAX-запиту, який надсилається на сервер кожного разу, коли користувач обирає нове місто у

переліку доступних, що випадає. У цю функцію серверний скрипт повертає єдиний рядок, який містить усі температури, що спостерігалися у даному місті, розділені крапкою з комою (як у форматі CSV). Ця функція розбиває рядок на окремі значення температури і записує їх у відповідні поля;

– ChangedCitySelect() – функція-обробник зміни списку, що випадає, яка ініціює AJAX-запит і приєднує до нього одне єдине поле, що містить назву міста, для якого слід завантажити температури;

– bindhandlers() – стандартна для JavaScript функція, яка викликається уже коли всі елементи управління створені, і, отже, для них можна задати обробники методом addEventListener.

Ще однією особливістю програмної реалізації є використання бази даних для збереження інформації про температури деяких «стандартних» міст. Структура цієї бази даних наведена на рис. 3.9.

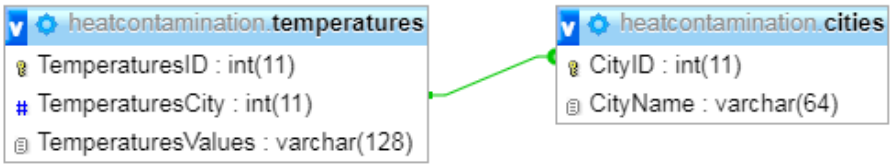

Рисунок 3.9 – Структура бази даних для збереження температурної інформації про деякі «стандартні» міста

До бази даних виконується два види запитів:

– `SELECT CityName FROM cities;` – для зчитування усього переліку міст, для яких є температури у базі, цей запит виконується один раз при завантаженні сторінки;

– `SELECT TemperaturesValues FROM temperatures,cities WHERE temperatures.TemperaturesCity=cities.CityID AND cities.CityName='".$_POST["city"]."';` – для отримання температур обраного міста, цей запит виконується під час кожного AJAX-запиту.

Інші аспекти програмної реалізації є більш-менш очевидними і тому можуть безпосередньо визначатися із вихідного тексту програмного продукту (див. Додаток Б).

3.4. Документаційне забезпечення розробленого програмного продукту

Розроблене програмне забезпечення є спеціалізованим інструментом і в основному призначене для фахівців, що займаються проблемами зміни клімату, хоча, звичайно, може бути використане і будь-яким пересічним користувачем мережі Інтернет. У будь-якому випадку до програмного забезпечення має поставлятися комплект документації, з якого обов'язковим елементом є інструкція користувача. Її можна скомпонувати із пунктів, що слідують нижче.

1) Для початку роботи з програмою зайти у браузер та задати адресу розробленого веб-додатку (локальну – при роботі на локальному сервері, або Інтернет-адресу – за умови, що розроблений продукт буде вміщено у відкритий доступ у Веб). Оскільки програмний продукт є розподіленим, має серверну частину, то простого клацання на файлі index.php буде недостатньо, його слід завантажувати саме через встановлений локальний або Інтернет-сервер.

2) Після завантаження сторінки користувач має для себе обрати режим роботи: із новим містом або з одним із міст, які знаходяться у базі даних. При роботі з новим містом перейти до пункту 4, а при роботі з базою даних перейти до наступного пункту.

3) У верхній частині сторінки, у списку, що випадає, обрати потрібне місто із загального переліку і перейти до пункту 6.

4) У верхній частині вікна задати кількість років p, за які в наявності у користувача є статистична інформація про температури у місті, для якого ведеться моделювання.

5) У горизонтально орієнтованій табличці задати усі середньорічні температури, що спостерігалися протягом p років у даному місті.

6) У вертикально орієнтованій табличці задати значення усіх показників міста, що моделюється. Якщо якийсь показник не відомий, то слід зняти прапорець біля його назви, щоби він таким чином не враховувався при розрахунках, а відповідне поле можна залишити порожнім (при знятому прапорці воно ігнорується).

7) Перевірити та, за необхідністю, змінити значення вагових коефіцієнтів, що приписані для кожного із показників, які будуть ураховуватися при розрахунку.

8) Коли усі дані будуть введені, слід ще раз усі їх перевірити та натиснути кнопку «Розрахувати!».

9) У новому вікні, яке відкривається після натискання на кнопку «Розрахувати!», слід подивитися та проаналізувати кінцевий результат. За необхідністю записати його на постійні носії.

10) Для завершення роботи з програмою слід закрити вікно браузера.

Керуючись цією інструкцією, користувач будь-якої кваліфікації зможе здійснювати моделювання теплового забруднення у наступний період.

3.5. Тестування системи та аналіз результатів її роботи

Після зведення програмного продукту було виконано його тестування (протягом кількох годин), протягом якого було промодельовано теплове забруднення більше, ніж десятка різних міст при різних значеннях поточних показників $\vec{u}_i$. Тестування показало, що програмне забезпечення працює без очевидних помилок, стабільно виконуючи задачу, поставлену у даній роботі.

Результати моделювання оцінювалися наступним чином:

– розраховувалося прогнозне значення температури для 2018 року тільки на основі традиційної регресійної моделі (без урахування показників $\vec{u}_i$) – для цього використовувалося те ж саме програмне забезпечення, але з ігноруванням усіх показників (усі прапорці було знято);

– розраховувалося прогнозне значення 2018 р. на основі регресійної моделі, скоригованої відповідно до запропонованої нової формули (3.8);

– обчислювалася ступінь відхилення обох спрогнозованих значень від реальної середньорічної температури, яка спостерігалася у 2018 р.

Результати порівняння показали майже на 15 % краще співпадіння результатів запропонованої моделі у порівнянні із традиційною регресією.

Як висновок, можна сказати, що розроблене програмне забезпечення може застосовуватися для більш точної оцінки температур майбутніх періодів у порівнянні із традиційними статистичними методами регресійного аналізу.

ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

У даній роботі виконано дослідження проблеми поширення забруднень на прикладі найбільш стратегічно важливого його типу – теплового забруднення, яке становить небезпеку для існування всієї людської цивілізації в цілому. Розповсюдження теплового забруднення розглядається у часі, оскільки просторові осередки його є очевидними і повністю співпадають із сучасними мегаполісами та агломераціями. Першочергово здійснено аналіз відповідної науково-технічної літератури та за основу обрано метод регресійного аналізу побудови одновимірної регресії. На основі синтезу цього статистичного та кваліметричних підходів створено математичну модель процесу зміни температури у певній місцевості, що, відповідно, враховує часову залежність зміни температури у даній місцевості, а також комплекс факторів, що непрямим чином впливають на зростання (чи зменшення) теплового забруднення (такі, як, наприклад, приріст чи депопуляція населення у регіоні, розвиток промислового виробництва, і т.п.).

Створена математична модель реалізована програмно у фронт-енд складовій за допомогою сучасної мови програмування JavaScript, яка очевидно є такою, що найбільш динамічно розвивається на сьогоднішній день серед усіх інших мов, в той час як необхідні для роботи додатку довідкові дані, завантажуються із бази даних (якою обрано MySQL) за допомогою коду, написаного на PHP. Використано популярне середовище розробки Microsoft Visual Studio Code (2019 р.). Зважаючи на специфіку програмного продукту, що має пакетний режим роботи, обрано структурну технологію програмування. Створене програмне забезпечення протестоване та встановлена ефективність його роботи, а саме точність прогнозування температури (основний числовий показник теплового забруднення) покращується до 15 %. Зважаючи на це, мету роботи можна вважати досягнутою, а саму роботу – завершеним дослідженням.

У перспективі робота може розширюватися шляхом залучення нових показників, що впливають на зростання температури, а також шляхом впровадження інших варіантів регресійної залежності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Біляєв М. М. Моделювання і прогнозування стану довкілля :підручник для студентів вищих навчальних закладів / М. М. Біляєв, В. В. Біляєва, П. С.; Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна МОН України. Кривий Ріг: Вид. Р. А. Козлов, 2016. 207 с.

2 Ковальчук П. І. Моделювання і прогнозування стану навколишнього середовища: Навч. Посібник. К.: Либідь, 2003. 208 с.

3 Попов О. О. Математичне моделювання розповсюдження техногенного забруднення від підприємств паливної енергетики: зб. наук. праць Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України. К.: ІПМЕ ім. Г.Є. Пухова НАН України, 2009. Вип. 51. С. 73-84.

4 Бідюк П.І., Савенков О.І., Трофимчук О.М.. Моделювання забруднення атмосфери та поверхневих вод в Україні. Екологічна безпека та природокористування, 2011 р. С.16-34.

5 Жигайло О.Л. Моделювання антропогенного забруднення грунтово-рослинного покриву та методи контролю: конспект лекцій. Дніпропетровськ: «Економіка», 2005. 107 с.

6 Сафранов Т.А., Польовий А.М., Коніков Є.Г., Ротар М.Ф., Ільїна В.Г., Паєвська Л.А., Сучков І.О. Антропогенне забруднення геологічного середовища та грунтово-рослинного покриву: навч. посіб. Одеса, Вид-во “ТЭС”, 2003. 260 с.

7 Bengio Y., Lamblin P., Popovici D., and Larochelle H.. Greedy layer-wise training of deep networks, in Proc. Neural Inf. Process. Syst., Cambridge, MA, USA, 2007.pp. 153–160.

8 Gangwar R. K., Mathur A. K., Gohil B. S., and Sujit Basu. Neural network based retrieval of atmospheric temperature profile using AMSU-A observations, International Journal of Atmospheric Sciences, vol. 2014, Article ID 763060, 8 p.

9 Raptis C. E., van Vliet M. T. H., Pfister S. Global thermal pollution of rivers from thermoelectric power plants // Environmental Research Letters. 2016. Vol. 11, No. 10. P. 104011. URL: https://pure.iiasa.ac.at/id/eprint/13989/1/erl_11_10_104011.pdf (дата звернення: 15.06.2025). pure.iiasa.ac.at

10 Caissie D. The thermal regime of rivers: a review // Freshwater Biology. 2006. Vol. 51, No. 8. P. 1389–1406. URL: https://www.waterboards.ca.gov/waterrights/water_issues/programs/bay_delta/california_waterfix/exhibits/docs/Stockton/part2rebuttal/stkn_64.pdf (дата звернення: 15.06.2025). waterboards.ca.gov

11 Issakhov A., Zhandaulet Y., Nogaeva A., Bekezhanova V. Mathematical Modelling of Thermal Process to Aquatic Ecosystem from Thermal Power Plant // The Scientific World Journal. 2014. Article ID 497368. P. 1–12. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC4060168/ (дата звернення: 15.06.2025). PMC

12 Davis L. R., Shirazi M. A. A Review of Thermal Plume Modeling. Corvallis: U.S. EPA, 1977. 132 p. URL: https://nepis.epa.gov/Exe/ZyPURL.cgi?Dockey=940032WX.TXT (дата звернення: 15.06.2025). nepis.epa.gov

13 Miara A., Vörösmarty C. J., Stewart R. J., Wollheim W. M., Rosenzweig B. R. Thermal pollution impacts on rivers and power supply in the United States under climate change scenarios // Environmental Research Letters. 2018. Vol. 13, No. 12. P. 124011. URL: https://www.osti.gov/pages/biblio/1457125 (дата звернення: 15.06.2025). osti.gov

14 Piccolroaz S., Toffolon M., et al. Lake Water Temperature Modeling in an Era of Climate and Environmental Change: A Review // Reviews of Geophysics. 2024. Vol. 62, No. 3. e2023RG000816. URL: https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2023RG000816 (дата звернення: 15.06.2025). agupubs.onlinelibrary.wiley.com

15 Arismendi I., Johnson S. L., Dunham J. B., Haggerty R. Descriptors of natural thermal regimes in streams and their responsiveness to change in the Pacific Northwest of North America // Freshwater Biology. 2013. Vol. 58, No. 5. P. 880–894. URL: https://andrewsforest.oregonstate.edu/pubs/pdf/pub4791.pdf (дата звернення: 15.06.2025). Andrews Forest Research Program

16 Niu X., Chen J., Miao S., Ma Y., Wang Z. Numerical Modeling of Water Thermal Plumes Emitted by Power Plants in Coastal Areas. Water. 2016. Vol. 8, No. 11. P. 482. URL: https://www.mdpi.com/2073-4441/8/11/482 (дата звернення: 15.06.2025). MDPI

17 Winter J., Zhang K., Yang M., Zhang Y., Jozic S. Quantifying Thermal Discharges from Nuclear Power Plants: Monitoring and Modeling Approaches //

Applied Sciences. 2025. Vol. 15, No. 2. P. 738. URL: https://www.mdpi.com/2076-3417/15/2/738 (дата звернення: 15.06.2025). MDPI

18 Ahmed A. N., et al. The urban heat island effect: a review on predictive models, experiments, and mitigation strategies // Sustainable Cities and Society. 2025. (in press). URL: https://www.sciencedirect.com/science/article/pii/S2590252025000480 (дата звернення: 15.06.2025). ScienceDirect

19 Kong G., et al. Modelling urban heat island effects: a global analysis of drivers and predictive performance // Discover Sustainability. 2025. Vol. 5. Article 78. URL: https://link.springer.com/article/10.1007/s43762-025-00178-w (дата звернення: 15.06.2025). SpringerLink

20 Lu H., et al. Climate data for building simulations with urban heat island and future warming signals // Scientific Data. 2024. Vol. 11. Article 587. URL: https://www.nature.com/articles/s41597-024-03532-5 (дата звернення: 15.06.2025). Nature

21 Raptis C. E., van Vliet M. T. H., Pfister S. (supplementary context). Impacts of recent drought and warm years on water resources and power supply in Europe // Environmental Research Letters. 2016. Vol. 11, No. 12. P. 124021. URL: https://pure.iiasa.ac.at/id/eprint/14245/1/van_Vliet_2016_Environ._Res._Lett._11_124021.pdf (дата звернення: 15.06.2025). pure.iiasa.ac.at

22 Гребінь В. В. Сучасний водний режим річок України: ландшафтно-гідрологічний аналіз. Київ: Ніка-Центр, 2010. 312 с. URL: https://geo.knu.ua/wp-content/uploads/2021/03/10_n_lit_gidrol.pdf.pdf (дата звернення: 15.06.2025). Географічний факультет КНУ

23 Мкртчян О. С., Шубер П. М. Геоінформаційне моделювання температурного поля західних регіонів України. Фізична географія та геоморфологія. 2009. Вип. 56. С. 140–147. URL: https://geography.lnu.edu.ua/wp-content/uploads/2017/04/Temperinterp_Shuber_%D0%A4%D0%93%D0%9C_09.pdf (дата звернення: 15.06.2025). Географічний факультет

Додаток А

ПЕРЕЛІК КОПІЙ ДЕМОНСТРАЦІЙНОГО МАТЕРІАЛУ

Слайд 1 – Формалізація (постановка) задачі (формули (1.1), (1.7)-(1.12), с. 14-26).

Слайд 2 – Математичне вирішення задачі (формули (1.12), (3.8), с.26, с. 65).

Слайд 3 – Програмна реалізація (рис. 3.2, рис. 3.3, с. 53).

Слайд 4 – Алгоритми, застосовані у програмі (рис. 3.1, с. 52, рис. 3.7-рис. 3.8, с. 63-64).

Слайд 5 – Висновки та результати (с. 67).

Додаток Б

КОД ВЕБ-СТОРІНКИ, ЩО СТАНОВИТЬ ОСНОВУ ДЛЯ СИСТЕМИ МОДЕЛЮВАННЯ ПОШИРЕННЯ ТЕПЛОВОГО ЗАБРУДНЕННЯ МОВОЮ HTML+PHP+JAVASCRIPT

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Моделювання теплового забруднення місцевості</title>
<script>
const params=new Array("Чисельність населення, осіб","Густина населення, осіб/кв.км", "Відсоток приросту населення за рік, %", "Середня місячна зарплатня, у.о.", "Насиченість промисловими підприємствами (по 5-бальній шкалі)", "Наявність електростанцій (у 30 км зоні), кількість");

const startw=new Array(10,8,6,5,7,4);
const ro=new Array(1000000000,1000000,1000,100000,100,100);

let nump=10;

function filldiv()
{
     document.getElementById("nump").value=nump;
    paramlist=document.getElementById("paramlist");
    paramlist.innerHTML="";
    let str="<table border=\"0\"><tr><td><table border=1 cellpadding=5 cellspacing=20><td align=\"right\">Рік</td>";
    for(let i=0;i<nump;i++)
    {
        str+="<td align=\"center\">";
        str+=""+(2019-nump+i);
        str+="</td>";
    }
    str+="</tr><tr><td>Середньорічна температура, С</td>"
    for(let i=0;i<nump;i++)
```

```
    {
        str+="<td align=\"center\">";
        str+="<input type=\"text\" id=\"temp"+i+"\" size=3>";
        str+="</td>";
    }
    str+="</tr></table></td></tr><tr><td>"
    str+="<table border=1 cellpadding=5 cellspacing=20>"
    for(let j=0;j<params.length;j++)
    {
        str+="<tr><td
align=\"right\">"+params[j]+"</td><td><input    type=\"checkbox\"
id=\"chk"+j+"\"  checked  title=\"Зніміть  флажок,  якщо  даний
показник враховувати не потрібно\"></td>";
        str+="<td><select  id=\"sel"+j+"\"  title=\"Змініть,  якщо
вважаєте  за  потрібне,  вагу  даного  показника  задану  за
умовчанням\">";
        for(let i=0;i<nump;i++)
        {
            str+="<option";
            if(startw[j]==(i+1))
                str+=" selected";
            str+=">"+(i+1)+"</option>";
        }
        str+="</select></td>";
          str+="<td>";
          str+="<input   type=\"text\"   id=\"par"+j+"\"   size=5
class=\"numbertoright\">";
          str+="</td>";
        str+="</tr>";
    }
    str+="</table></td></tr></table>";
    paramlist.innerHTML=str;
     for(let i=0;i<nump;i++)

     document.getElementById("temp"+i).addEventListener('change',
checknum,false);
    for(let j=0;j<params.length;j++)

document.getElementById("par"+j).addEventListener('change',check
num,false);
}

function startcalc()
```

```
{
    let maindiv=document.getElementById("maindiv");
     let sT=0,st=0,stT=0,st2=0,Ti,ti,Ts,ts,wi,sw=0;
     let sum1=0,sum2=0,res,Tip1;
    for(let i=0;i<nump;i++)
    {
          Ti=+document.getElementById("temp"+i).value;
          ti=(2019-nump+i);
          sT+=Ti;
          st+=ti;
          stT+=Ti*ti;
          st2+=ti*ti;
     }
     ts=st/nump;
     Ts=sT/nump;
     sum1=(stT/nump-ts*Ts)*(2019-ts)/(st2/nump-ts*ts)+Ts;

     for(let j=0;j<params.length;j++)
     {
          ui=+document.getElementById("par"+j).value;
        wi=+document.getElementById("sel"+j).selectedIndex+1;
          sw+=wi;
          sum2+=ui/ro[j]*wi;
     }
     sum2/=sw;
     sum2+=1;
     res=sum1*sum2;


     let can=document.getElementById("mycanvas");
     let ctx=can.getContext("2d");
     ctx.strokeStyle="red";
     ctx.beginPath();
     ctx.lineTo(100,50);
     ctx.lineTo(100,400);
     ctx.lineTo(500,400);
     ctx.moveTo(90,60);
     ctx.lineTo(100,50);
     ctx.lineTo(110,60);
     ctx.moveTo(490,390);
     ctx.lineTo(500,400);
     ctx.lineTo(490,410);
     ctx.moveTo(90,100);
```

```
     ctx.lineTo(110,100);
     ctx.moveTo(90,200);
     ctx.lineTo(110,200);
     ctx.moveTo(90,300);
     ctx.lineTo(110,300);
     ctx.fillText("30",70,110);
     ctx.fillText("20",70,210);
     ctx.fillText("10",70,310);
     for(let i=0;i<nump;i++)
    {
          ctx.fillText(""+(2019-nump+i),85+Math.floor(i*(500-
100)/nump),420);
          ctx.moveTo(100+Math.floor(i*(500-100)/nump), 390);
          ctx.lineTo(100+Math.floor(i*(500-100)/nump), 410);
          Ti=+document.getElementById("temp"+i).value;
          if(i<nump-1)
               Tip1=+document.getElementById("temp"+(i+1)).value;
          else
               Tip1=res;
          ctx.moveTo(100+Math.floor(i*(500-100)/nump),       400-
Math.floor(Ti/40*400));
          ctx.lineTo(100+Math.floor((i+1)*(500-100)/nump),   400-
Math.floor(Tip1/40*400));
     }
     ctx.fillText("Традиційна",520,400-
Math.floor(sum1/40*400),200);
     ctx.moveTo(100+Math.floor((nump-1)*(500-100)/nump),     400-
Math.floor(Ti/40*400));
     ctx.lineTo(100+Math.floor(nump*(500-100)/nump),         400-
Math.floor(sum1/40*400));
     ctx.fillText("Запропонована",520,400-
Math.floor(Ti/40*400),200);

     ctx.stroke();
     maindiv.innerHTML="Прогнозована                  температура:
"+res+"<br>Дякуємо за використання нашої системи!";
}

function checknum()
{
    const temp=+this.value;
    if(!temp)
    {
```

```
        this.value="";
        alert("Вводіть, будь-ласка, тільки числові дані!");
    }
    else
    {
        if(this.id=="nump")
        {
            nump=temp;
            filldiv();
        }
    }
}

function filltemperatures(e)
{
     var T=e.target.responseText.split(';');
     nump=T.length;
     filldiv();
     for(let i=0;i<nump;i++)
          document.getElementById("temp"+i).value=T[i];
}

function ChangedCitySelect()
{
     var index=document.getElementById("city").selectedIndex;
     CitySelected=document.getElementById("city").options[index].
text;
     var data=new FormData();
     data.append('city',CitySelected);
     var req=new XMLHttpRequest();
     req.addEventListener('load',filltemperatures,false);
     req.open("POST","getT.php",true);
     req.send(data);
}

function bindhandlers()
{

document.getElementById("docalc").addEventListener('click',start
calc,false);

document.getElementById("nump").addEventListener('change',checkn
um,false);
```

```
     document.getElementById("city").addEventListener('change',Ch
angedCitySelect,false);
    filldiv();
}

window.addEventListener('load',bindhandlers,false);
</script>
<link rel="stylesheet" href="mycss.css">
</head>
<body>
<div id="maindiv">
<form>
Введіть кількість попередніх років, для яких є значення середніх
температур:<br/>p = <input     type="text"     id="nump"
size="8" value="10"><br><br>
Можна вибрати одне із міст, по яким наявні температури, внесені
у базу даних:<br/>
<select id="city">
<option>Оберіть місто...</option>
<?php
$hostname="localhost";
$username="root";
$password="";
$dbName="heatcontamination";
$link=mysql_connect($hostname,$username,$password)   or   die("No
connect to DB!");
mysql_set_charset('utf8',$link);
mysql_select_db("$dbName",$link) or die("Cannot select DB!");
$query="select CityName from cities;";
$result=mysql_query($query);
$num_rows=mysql_num_rows($result);
for($i=0;$i<$num_rows;$i++)
{
     $row=mysql_fetch_array($result);
     print     "<option>".$row['CityName']."</option>";
}
?>
</select>
<div id="paramlist">
</div>
<input type="button" value="Розрахувати!" id="docalc">
</form>
</div>
```

```
<canvas id="mycanvas" width="600" height="500"></canvas>
</body>
</html>
```

Додаток В

КОД СКРИПТА, ЩО ЗДІЙСНЮЄ ВІДПОВІДЬ НА AJAX-ЗАПИТИ ВІД ВЕБ-СТОРІНКИ

```
<?php
header('Content-Type: text/html; charset=UTF-8');
$hostname="localhost"; $username="root"; $password="";
$dbName="heatcontamination";
$link=mysql_connect($hostname,$username,$password)   or   die("No
connect to DB!");
mysql_set_charset('utf8',$link);
mysql_select_db("$dbName",$link) or die("Cannot select DB!");
$query="SELECT TemperaturesValues FROM temperatures,cities WHERE
temperatures.TemperaturesCity=cities.CityID                    AND
cities.CityName='".$_POST["city"]."';";
$result=mysql_query($query);
$num_rows=mysql_num_rows($result);
for($i=0;$i<$num_rows;$i++)
{
     $row=mysql_fetch_array($result);
     print $row['TemperaturesValues'];
     if($i!=$num_rows-1)
          print ";";
}
?>
```

Додаток Г

КОД КАСКАДНОЇ ТАБЛИЦІ СТИЛІВ, ЗАСТОСОВАНОЇ ДЛЯ РОЗРОБЛЕНОЇ ВЕБ-СТОРІНКИ НА CSS

```
table, table td
{
    color:darkblue;
    background-color: lightblue;
    border-radius: 10px;
    font-weight: bold;
}

input[type="text"], select
{
    background-color: rgb(230, 245, 255);
}

input[type="button"]
{
    width:400px;
    height:75px;
    border-radius: 10px;
    font-size: 28pt;

}
```

Додаток Д

ЗМІСТ БАЗИ ДАНИХ, ЩО ВИКОРИСТАНА В РОБОТІ

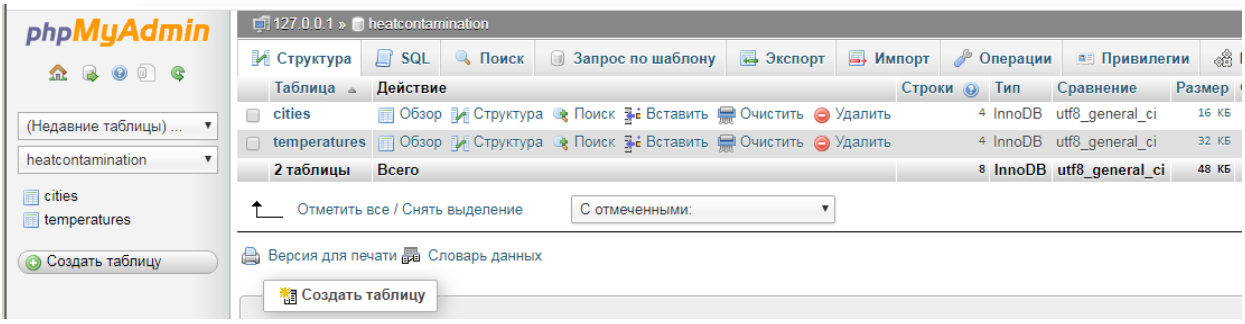

Рисунок Д.1 – Перелік таблиць бази даних heatcontamination

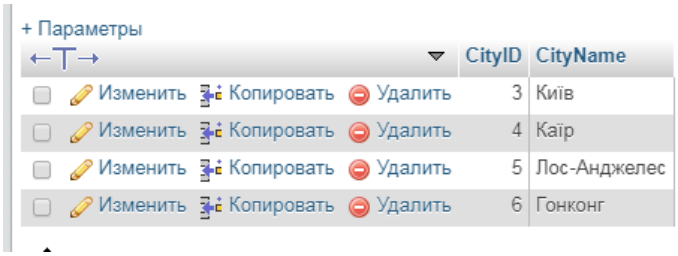

Рисунок Д.2 – Вміст таблиці cities

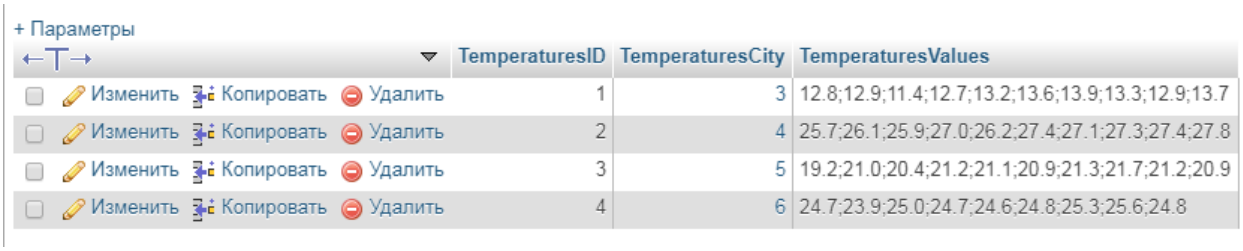

Рисунок Д.3 – Вміст таблиці temperatures