ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет електроніки
Кафедра фізики напівпровідників

Пояснювальна записка

до дипломної роботи
бакалавр
(освітньо-кваліфікаційний рівень)

на тему Програмна реалізація методу PERT управління проектами

Виконав: студент 4 курсу, групи ФЕІ-42
напряму підготовки (спеціальності)
6.050101 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Гопчак Н. І. _____
(прізвище та ініціали)
Керівник Ненчук Т. М. _____
(прізвище та ініціали)
Рецензент Катеринчук І. М. _____
(прізвище та ініціали)

Львів - 2014 року

Анотація

В дипломній роботі розглянуто методи PERT (Project Evaluation and Review Technique) і CPM (Метод критичного шляху) управління проектами.

Описано середовище програмування Microsoft Visual Studio, мову C# та СУБД Microsoft Access.

Реалізовано повнофункціональний метод PERT, створено зручний користувацький інтерфейс. Проведено тестування роботи програми.

Annotation

In the graduate work it was conside PERT (Project Evaluation and Review Technique) and CPM (Critical Path Method) project management methods.

We describe a programming environment Microsoft Visual Studio, C # language and DBMS Microsoft Access.

It was implemented a fully functional PERT method, created a convenient user interface, conducted program testing .

Зміст

Вступ

У системі управління проектом важливе місце займає планування - розробка системи цілеспрямованих дій з реалізації інвестиційного проекту, що передбачає порядок, послідовність і терміни виконання робіт і забезпечує ефективне використання матеріально-технічних, трудових і фінансових ресурсів. Прийняті на основі планів управлінські рішення мають відповідати прогресивним організаційно-технічним і технологічним принципам здійснення всіх видів робіт в задані терміни і з високою їх якістю.

Планування охоплює всі фази здійснення проекту на різних стадіях і різниться лише специфікою планованих робіт і складністю їх виконання. Разом з тим існують загальні, властиві всім стадіям проекту, підходи. Суть їх полягає в припущенні, що цілі проекту та обсяги робіт, необхідні для їх досягнення, можуть бути структуровані й описані досить точно в термінах і оцінках. З іншого боку прийняті за основу способи реалізації поставлених у проекті цілей визначають послідовність виконання окремих видів і етапів робіт, кваліфікацію виконавців, від яких залежать строки здійснення проекту і його вартість.

Для великої кількості проектів тривалість виконання всіх або деяких робіт неможливо визначити точно. Перш за все мова йде про проектування і впровадження нових систем. У таких проектах багато робіт не мають аналогів. В результаті виникає невизначеність у термінах виконання проекту в цілому.

Мета дипломної роботи полягає у створенні програми для реалізації повнофункціонального методу PERT (Project Evaluation and Review Technique). Актуальність дипломної роботи полягає у тому, що у сучасних популярних програмних середовищах управління проектами метод PERT не реалізований взагалі, або реалізований частково, і не виконує свого основного призначення – прогнозування ймовірності виконання проекту за деякий заданий час.

1. Управління проектами : основні поняття

1.1. Поняття проекту

Проект — це обмежений часовими рамками процес, що має визначений початок та кінець, зазвичай обмежений датою, але також може обмежуватися фінансуванням або досягненням результатів, який здійснюється для реалізації унікальних цілей та завдань, зазвичай, щоб призвести до вигідних змін або створення доданої вартості. Тимчасова природа проектів контрастує з бізнесом (процесами), які є повторюваною, постійною або частково постійною діяльністю з виробництва продуктів або послуг. На практиці, управління вищезазначеними двома системами часто різниться і таким чином вимагає розвитку окремих технічних навичок та використання розподіленого управління ними. Проект – це унікальний комплекс взаємозв’язаних дій, які напрямлені на досягнення конкретної мети за умови визначених вимог до термінів бюджету і характеристик отриманих результатів.

Дії і проекти в основному відрізняються тим, що перші виконуються весь час і є чимось постійним, тоді як проекти є тимчасовими і унікальними. Отже, проект може бути визначений через свої характеристики: проект - це тимчасова дія, що виконується для створення унікального продукту чи послуги.

Під проектом розуміють комплекс науково-дослідних, проектно-конструкторських, соціально-економічних, організаційно-господарських та інших заходів, пов'язаних ресурсами, виконавцями та термінами, відповідно оформлених і направлених на зміну об'єкта управління, що забезпечує ефективність розв'язання основних завдань та досягнення відповідних цілей за певний період. Кінцевими цілями проектів є створення та освоєння нової техніки, технології та матеріалів, що сприяє виходу продукції на світовий рівень.

Можна навести ще кілька варіантів визначення поняття "проект", які зустрічаються в літературі:

Проект - це окреме підприємство з конкретними цілями, які часто

включають вимоги до часу, вартості та якості результатів, що досягаються (Англійська асоціація проект-менеджерів).

Проект - це певне завдання з визначеними вихідними даними й встановленими результатами (цілями), що обумовлюють спосіб його вирішення (Тлумачний словник з управління проектами).

Ці визначення є універсальними, методологічно виваженими та широко застосовуваними в зарубіжній практиці управління проектами.

В ринкових умовах організація проекту спрямована на те, щоб у рамках існуючого підприємства вирішити: одиночну, інноваційну і тому ненадійну, обмежену в часі, комплексну задачу.

Для успішного здійснення проекту необхідно виділити його основні ознаки, які дозволять менеджерам використати необхідний інструментарій для реалізації проекту. Основні ознаки проекту наведено на Рис. 1. Зокрема, кількісне вимірювання означає, що оцінка проекту буде здійснюватися за числовими величинами, що описують результати проекту. А життєвий цикл охоплює всі фази розвитку проекту у часі.

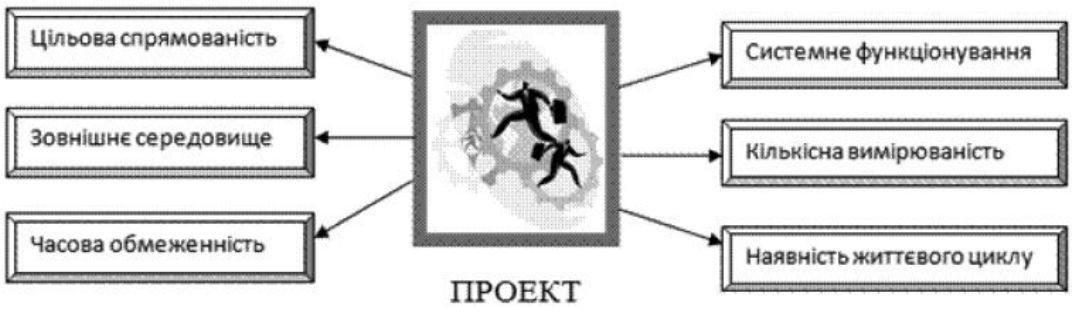


Рис. 1. Основні ознаки проекту

План — це фіксація системи цілей, задач і засобів, які передбачають спрямовану зміну ситуації при передбаченому стані середовища.

Програма — це сукупність взаємопов’язаних проектів (які виконувались у минулому, тих, які виконуються сьогодні та запланованих), а також комплекс організаційних змін, об’єднаних загальними цілями і спрямованих на досягнення конкретної комерційної вигоди.

Портфель проектів — множина проектів, програм та інших робіт, які

виконуються у даний час і об’єднані разом з метою ефективного управління для досягнення стратегічних цілей організації.

Стратегічний портфель — збалансована сукупність проектів, які динамічно змінюються та спрямовані на реалізацію стратегічних цілей компанії. Стратегічний портфель організації являє собою набір діючих програм, проектів, підлеглих портфелів та інших робіт компанії в певний момент часу. Послідовність проектів називається ланцюжком проектів, який не є портфелем, але цілком може бути програмою.

1.2. Суть управління проектами

Управління проектами — область знань з планування, організації та управління ресурсами з метою успішного досягнення цілей та завершення завдань проекту; область IT менеджменту, яка охоплює ті сфери діяльності, в яких створення продукту або послуги реалізується як унікальний комплекс загальнозв’язаних цілеспрямованих подій, за умови певних вимог до термінів, бюджету і характеристик отриманих результатів.

Суть проектного менеджменту полягає в управлінні цілями організації, що дозволить компанії бути успішною в конкурентній боротьбі, швидко реагувати на зовнішні і внутрішні зміни, заощаджувати час і гроші. Саме ці три моменти: час, бюджет і якість робіт знаходяться під постійною увагою керівника проекту.

Проектна тріада (час, бюджет і якість робіт) є основними обмеження, що накладаються на проект.

Відповідно до загальноприйнятих принципів управління проектами вважається, що ефективне управління термінами робіт є ключем до успіху усіх трьох показників. Часові обмеження проекту є найкритичнішими. Там, де терміни виконання проекту суттєво затягуються, дуже ймовірними наслідками є перевитрата засобів і недостатньо висока якість робіт. Тому у більшості методів управління проектами основний акцент робиться на календарному плануванні робіт і контролі за дотриманням календарного графіка. З трьох основних обмежень найскладніше контролювати якість

заданих результатів проекту.

Управління проектом — це діяльність, спрямована на реалізацію проекту з максимально можливою ефективністю при заданих обмеженнях за часом, коштами (і ресурсами), а також якості кінцевих результатів проекту.

Суб'єкт управління – управлінська організаційна структура (орган, підрозділ органу, трудовий колектив, службовець, громадська організація тощо), що включена в певну систему, підсистему управління і наділена спеціальною компетенцією і необхідними ресурсами, яка здійснює цілеспрямовану діяльність щодо забезпечення виконання соціально-економічних завдань даної системи, підсистеми.

Об'єкти управління - елементи структури системи управління та виробничі процеси, на які спрямований вплив функцій управління.

Головним завданням проектного управління є досягнення всіх цілей та виконання завдань проекту, одночасно виконуючи зобов'язання щодо наперед визначених обмежень проекту. Другорядним завданням, але амбіційнішим, є оптимізація, розподілення та інтеграція завдань, необхідних для досягнення наперед визначених цілей. Проект передбачає наявність плану досягнення поставлених цілей (комплексу робіт), а також наявність системи повноважень і відповідальності за досягнення цілей на чолі з менеджером проекту.

Статут проекту - документ, який відображає ключові відомості про проект і розробляється в ході підготовки проекту. Завдання даного документа - сформувати єдине уявлення про проект у всіх учасників і формалізувати досягнуті домовленості.

1.3. Стратегія, мета та цілі проекту

Планування цілей проекту (Scope Planing). Розробка документа, у якому визначені цілі проекту. Відправною точкою слугують опис продукту, обґрунтування проекту, загальні обмеження, інформація про уже виконані аналогічні проекти. Аналізуються альтернативні шляхи реалізації проекту, визначаються критерії успішності. Цей документ надалі є основою для усіх

проектних рішень і єдиного розуміння цілей проекту всіма його учасниками.

Місія - це генеральна ціль проекту, причина його існування. Вона визначає орієнтири для наступних рівнів цілей, а також для розробки стратегії на різних організаційних рівнях.

Стратегія проекту — це спільне бачення шляху досягнення цілей.

Цілі проекту — бажані результати дій, що вирішують поставлену проблему, і які повинні бути досягнуті в ході реалізації проекту.

Зазвичай на практиці виділяють три рівні пріоритетів цілей проектів:

1. Основні цілі проекту. Мають бути досягнуті для того, щоб проект вважався успішно реалізованим.

2. Необхідні цілі. Їх потрібно досягти в ході реалізації проекту, проте при виникненні ускладнень ними можна частково пожертвувати.

3. Бажані цілі. Їх було б бажано досягти при здійсненні проекту.

Мета проекту - це бажаний і доведений результат, досягнутий у межах певного строку при заданих умовах реалізації проекту.

Досягнення поставленої мети вимагає розв'язання певних завдань, а саме:

- визначити можливі результати проекту (прогнозування);

- надати кількісну оцінку цим результатам;

- обґрунтувати можливість досягнення цих результатів та їх ефективність;

- визначити умови, за яких ці результати мають бути досягнуті.

Декомпозиція цілей (Scope Definition). Послідовний розподіл основних результатів проекту на більш дрібні елементи, аж до пакетів робіт, що добре піддаються керуванню. У результаті утворюється ієрархічна структура (дерево) робіт проекту (Work Breakdown Structure - WBS).

1.4. Класифікація проектів

Різноманітність проектів, які зустрічаються, можна класифікувати за різними критеріями (рис. 2):

– за класом проекту (складом і структурою самого проекту та його

предметної галузі) — монопроект — окремий проект різних типів, видів та масштабів; мультипроект — комплексний проект, який складається з ряду монопроектів і потребує застосування багатопроектного управління; мегапроект — цільові програми розвитку регіонів, галузей, держави, які включають до свого складу ряд моно- і мультипроектів;

– за типом проекту (основними сферами діяльності, в яких здійснюється проект) — технічні, організаційні, економічні, соціальні, змішані;

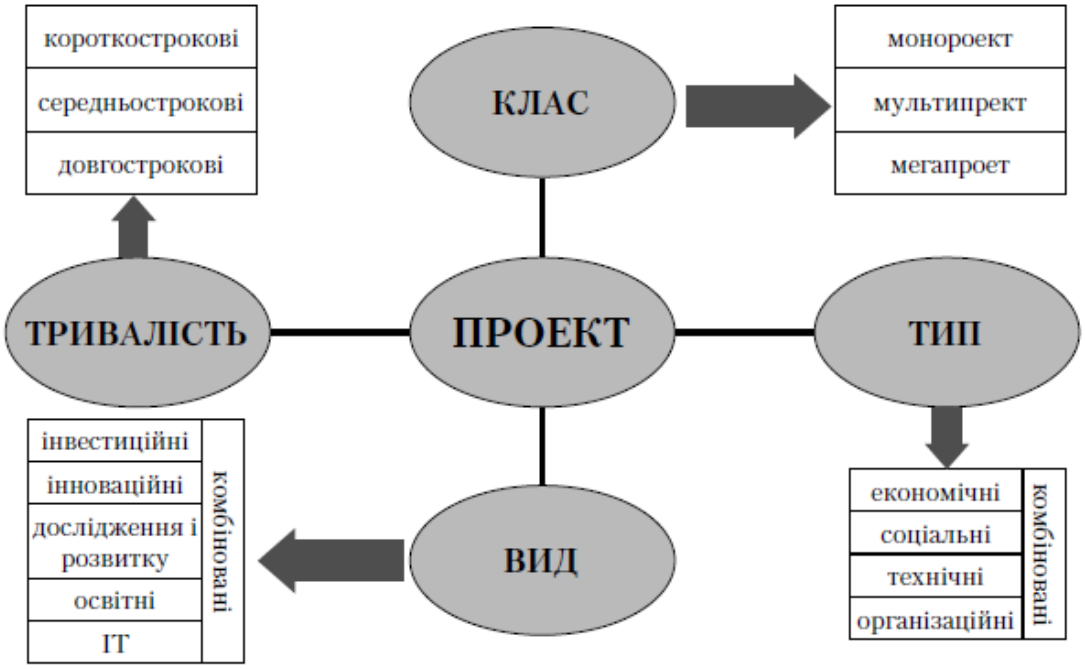


Рис. 2. Класифікація проектів за основними критеріями

– за видом проекту (характером предметної галузі проекту) — інвестиційні, інноваційні, дослідження і розвитку, освітні, ІТ, комбіновані. До інвестиційних зазвичай належать проекти, головною ознакою яких є створення чи реновація основних фондів, що вимагає вкладення інвестицій. До інноваційних проектів належать проекти, з яких головна мета полягає в розробці й застосуванні нових технологій, ноу-хау та інших нововведень, що забезпечує розвиток систем;

– за тривалістю проекту — короткострокові (до 3 років), середньострокові (від 3 до 5 років), довгострокові (понад 5 років).

1.5. Структура, оточення та учасники проекту

Контекст проекту - це зовнішнє та внутрішнє середовище в якому планується та здійснюється проект.

Для того щоб управляти проектом, доцільно розбити його на ієрархічні підсистеми та компоненти, тобто структурувати. Структура проекту - це організація зв'язків і відносин між його елементами. За допомогою структури визначають, що необхідно розробити чи зробити; вона пов'язує роботи між собою та з кінцевою метою проекту. У процесі структурування виокремлюють компоненти продукції проекту, етапи його життєвого циклу та елементи організаційної структури. Структурування - невіддільна частина загального процесу планування проекту, визначення його цілей, розподілу відповідальності й обов'язків. До основних завдань структурування проекту належать такі:

- поділ проекту на блоки, що підлягають управлінню;
- розподіл відповідальності за елементи проекту і визначення зв'язку робіт зі структурою організації (ресурсами);
- точне оцінювання необхідних витрат (коштів, часу і матеріальних ресурсів);
- створення єдиної бази для планування, упорядкування кошторисів і контролю за витратами;
- встановлення зв'язку між роботами, пов'язаними з проектом і системою ведення бухгалтерських рахунків;
- перехід від загальних, не завжди конкретно виражених цілей до конкретних, які виконують підрозділи організації;
- окреслення комплексів робіт (підрядів).

Мистецтво поділу проекту на складові полягає в умінні поєднувати три різні структури - процес, продукт і організація - в єдину структуру проекту.

Необхідно чітко окреслити характер, мету і зміст проекту, а також усі його кінцеві продукти з їх точними характеристиками. Доцільно здійснити ієрархію цілей, що показує повний ланцюг кінцевих результатів або засобів їх

досягнення. При цьому необхідно обміркувати потрібний рівень деталізації планів і оцінити кількість рівнів у структурі проекту.

Слід побудувати схему життєвого циклу проекту та організаційну схему, де зазначити групи чи окремих осіб, які працюватимуть над проектом, включаючи зацікавлених у проекті осіб із зовнішнього середовища проекту. Необхідно проаналізувати структуру продукції - схему її поділу на підсистеми чи компоненти, включаючи машини і устаткування, програмне та інформаційне забезпечення, послуги, а також у разі потреби географічний поділ. Крім того, потрібно вивчити план бухгалтерських рахунків організації - систему застосовуваних при структуруванні проекту кодів, яка має ґрунтуватися на діючому в організації плані бухгалтерських рахунків або з урахуванням його коригування. На основі отриманої інформації потрібно скласти генеральний зведений план проекту, який можна буде деталізувати у процесі пошуку критичного шляху.

При реалізації проекту цей план можна використовувати для доповідей керівництву. На основі зведеного плану слід скласти робочий план бухгалтерських рахунків (у разі потреби доцільно розробити систему субрахунків), робочий сітковий (мережевий) графік з часовими й ресурсними оцінками всіх робіт, а також запровадити систему завдань.

Оточення проекту (Project Environment) — сукупність зовнішніх та внутрішніх сил, які сприяють чи заважають досягненню цілей проекту.

Оточення є важливим елементом проекту, оскільки важливо визначити середовище, в якому виникає, існує і завершується проект. Оточення проекту - це чинники впливу на його підготовку та реалізацію. Тому їх можна поділити на внутрішні й зовнішні.

До зовнішніх належать політичні, економічні, суспільні, правові, науково-технічні, культурні та природні.

До політичних чинників належать: політична стабільність, підтримка проекту державними установами, міжнаціональні взаємини, рівень злочинності, міждержавні стосунки тощо.

До правових - стабільність законодавства, дотримання прав людини, прав власності, прав підприємництва.

До економічних - структура внутрішнього валового продукту, умови регулювання цін, рівень інфляції, стабільність національної валюти, розвиненість банківської системи, стан ринків, рівень розвитку підприємництва і т.п. Важливим при визначенні оточення проектів є рівень розвитку фундаментальних та прикладних наук, рівень інформаційних та промислових технологій, рівень розвитку енергетики, транспорту, зв'язку, комунікацій тощо.

До внутрішніх належать чинники, пов'язані з організацією проекту. Організація проекту є розподілом прав, відповідальності та обов'язків між учасниками проекту.

Учасниками управління проектами є юридичні або/та фізичні особи, які зобов'язані виконати деякі дії, що передбачені проектом, та інтереси яких будуть задіяні при реалізації проекту.

До числа учасників можуть входити інвестори, банки, підрядчики, постачальники, гуртові покупці продукції, лізингодавці та інші фізичні чи юридичні особи. Учасником проекту може бути також держава.

Автором головної ідеї проекту, його попереднього обґрунтування є ініціатор проекту. Ділова ініціатива у здійсненні проекту, як правило, належить замовнику.

Замовник - це зацікавлена сторона в здійсненні проекту, майбутній власник та користувач результатів проекту. Він визначає основні вимоги та масштаби проекту, забезпечує фінансування проекту за рахунок власних коштів або коштів інвесторів, укладає угоди з виконавцями проекту, несе відповідальність за ці угоди та в цілому за проект перед суспільством та законом, керує процесом взаємодії між учасниками проекту.

Якщо інвестор, тобто та сторона проекту, яка забезпечує його фінансування, не є замовником, то вкладення коштів у проект можуть здійснювати банки, інвестиційні фонди та інші кредитні організації.

Вони вступають у договірні відносини із замовником, контролюють виконання контрактів, здійснюють розрахунки з іншими сторонами по мірі виконання робіт. Ціллю інвесторів є максимізація прибутку зі своїх інвестицій від реалізації проекту. Вони є повноцінними партнерами проекту й власниками всього майна, придбаного за рахунок інвестицій, до того часу, поки не будуть виплачені всі кошти по контракту (кредитному договору) із замовником.

Свої повноваження по керівництву роботами зі здійснення проекту, а саме: планування, контроль та координацію робіт всіх учасників проекту, замовник та інвестор делегують керівнику проекту.

Склад функцій та повноважень керівника проекту визначається контрактом із замовником. Перед керівником та його командою ставиться завдання управління та координації робіт протягом життєвого циклу проекту для досягнення поставлених цілей та результатів при дотриманні встановлених термінів, бюджету та якості.

Проектно-кошторисну документацію розробляють проектні організації – проектувальники. Організацію, яка несе відповідальність за виконання комплексу проектних робіт, називають генеральним проектувальником.

Архітектор - це особа чи організація, що має право на основі відповідно оформленої ліцензії професійно виконувати роботу зі створення проектно-кошторисної документації, специфікацій, вимог до проведення тендерів (торгів), а також здійснювати загальне управління проектом.

Інженер - це особа чи організація, що має право на основі ліцензії займатися так званим інжинірингом - комплексом послуг, пов'язаних з процесом виробництва й реалізації продукції проекту. Інжиніринг передбачає планування робіт, інженерне проектування, здійснення випробувань, а також контроль за здачею об’єкта в експлуатацію.

Постачальник - це організація, що здійснює ресурсне забезпечення проекту (закупівлі та поставки).

Підрядчик (генеральний підрядчик, субпідрядник) - це юридична

особа, яка несе відповідальність за виконання робіт відповідно до контракту.

Консультант – це фірма чи спеціаліст, який на контрактних умовах надає учасникам проекту консультаційні послуги з питань його реалізації

Команда проекту - це специфічна організаційна структура, яку очолює керівник проекту. Вона створюється на період здійснення проекту і завданням її є здійснення функцій управління проектом.

Склад команди залежить від характеристик проекту, а саме від його масштабу, складності.

Членами команди є: інженер проекту, керівник контрактів, контролер проекту, бухгалтер проекту, керівник відділу матеріально-технічного забезпечення, керівник робіт із проектування, керівник виробництвом (будівництвом), адміністративний помічник.

Крім того, учасниками проекту є: контрактор або генеральний контрактор (сторона, яка бере на себе відповідальність за виконання робіт по контракту), субконтрактор (вступає в договірні відносини з контрактором чи субконтрактором більш високого рівня), координатор робіт з експлуатації, проектувальник (юридична особа, що виконує за контрактом проектно-дослідницькі роботи в межах проекту), генеральний підрядчик (юридична особа, чия пропозиція прийнята замовником, несе відповідальність за виконання робіт відповідно до умов контракту), ліцензори (організації, що виділяють ліцензії на право володіння земельною ділянкою, проведення торгів, виконання окремих робіт тощо), постачальники, органи влади, власник земельної ділянки, виробник кінцевої продукції проекту, споживачі продукції.

На здійснення проекту можуть впливати й інші сторони з оточення проекту, які можуть бути віднесені до учасників проекту, це: конкуренти основних учасників проекту, спонсори проекту, різні консалтингові, юридичні, посередницькі організації, залучені до процесу здійснення проекту.

1.6. Життєвий цикл проекту

Життєвий цикл проекту – це час від першої затрати до останньої вигоди проекту. Він відображає розвиток проекту, роботи, які проводяться на різних стадіях підготовки, реалізації та експлуатації проекту. Стадії життєвого циклу проекту можуть розрізнятися в залежності від сфери діяльності і прийнятої системи організації робіт. До поняття циклу проекту входить визначення різних стадій розробки й реалізації проекту. Цикл проекту являє собою певну схему або алгоритм, за допомогою якого відбувається встановлення певної послідовності дій при розробці та впровадженні проекту.

Життєвий цикл проекту визначає:

- яка робота має бути зроблена на кожній фазі;
- хто має бути залучений до проекту у кожній фазі.

Перша стадія циклу – ідентифікація – стосується вибору або генерування таких ґрунтовних ідей, які можуть забезпечити виконання важливих завдань розвитку. На цій стадії слід скласти перелік усіх можливих ідей, придатних для досягнення цілей економічного розвитку

Після того, як проект пройшов першу стадію циклу, необхідно прийняти рішення, чи варто продовжувати розгляд ідеї.

Стадія розробки. Для цього потрібне послідовне уточнення проекту за всіма його параметрами, а саме за його технічними характеристиками, врахування його впливу на довколишнє середовище, ефективності та фінансової здійснимості, прийнятності з соціальних і культурних міркувань, а також масштабності організаційних заходів.

Експертиза забезпечує остаточну оцінку всіх аспектів проекту перед запитом чи рішенням про його фінансування. На стадії експертизи увага, як правило, зосереджується на оптимальному варіанті.

На стадії переговорів інвестор і замовник, який хоче одержати фінансування під проект, докладають зусиль для того, щоб дійти згоди щодо заходів, необхідних для забезпечення успіху проекту.

Під реалізацією проекту розуміють виконання необхідних робіт для

досягнення його цілей. На стадії реалізації провадиться контроль і нагляд за всіма видами робіт чи діяльності в міру розвитку проекту. Порядок проведення контролю та інспекції має бути погоджено на стадії переговорів.

На стадії завершальної оцінки визначається ступінь досягнення цілей проекту, із набутого досвіду робляться висновки про його використання в подальших проектах. У перебігу цієї стадії треба порівняти фактичні результати проекту із запланованими.

Загальні властивості фаз життєвого циклу полягають у наступному:

1) матеріальні витрати та кількість залученого до проекту персоналу спочатку низькі, згодом зростають і швидко йдуть угору, коли проект наближається до завершення;

2) імовірність успішного завершення проекту є найменшою, а ризик і невизначеність відповідно найвищими на початку проекту; імовірність успішного завершення проекту поступово зростає в міру виконання проекту;

3) здатність зацікавлених осіб впливати на остаточні властивості продукту проекту і на остаточну вартість проекту найвища на початку виконання проекту, але з часом поступово стає нижчою.

2. Мережеве планування

Мережеве планування (network planning) – метод наукового планування та управління виробничими процесами; одна з форм графічного відображення змісту робіт і тривалості виконання стратегічних планів і довгострокових комплексів робіт.

Мережеве планування має ряд переваг: забезпечує наочність технологічної послідовності робіт; дозволяє скласти оперативні та поточні плани, а також прогнозувати складні процеси; дозволяє виявити приховані ресурси часу і матеріальних засобів при виконанні виробничих процесів.

Щоб приступити до мережевого планування (моделювання) того чи іншого виробничого процесу необхідно мати перелік і тривалість виконання операцій цього процесу. Мережеве планування супроводжується побудовою робочих таблиць і мережевих графіків.

Технологія мережного планування, що є основою систем управління проектами, складається з наступних методів: метод діаграм Ганта і мережні методи планування.

Діаграма Ганта представляє собою лінійний графік, що задає моменти часу початку і закінчення взаємозалежних дій, що складають єдиний технологічний процес, який потрібно виконати для досягнення мети проекту. Основними недоліками методу діаграм Ганта є складність формалізації процедур аналізу та відсутність можливості встановлення залежностей між різними подіями.

До позитивних якостей варто віднести простоту і наочність. Сучасні системи планування використовують модифіковані діаграми Ганта, у яких перелічені вище недоліки більшою мірою усунуті.

Мережевий графік - це динамічна модель виробничого процесу, що відображає технологічну залежність і послідовність виконання комплексу робіт, погоджує їх звершення в часі з урахуванням витрат ресурсів і вартості робіт з виділенням при цьому вузьких (критичних) місць. Основні елементи мережевого графіка - робота і подія.

Мережевий графік будується у вигляді графа, який відображає роботи проекту, зв'язки між ними, стан проекту.

Граф може бути побудований у двох варіантах:

а) вершини графа відображають стан деякого об'єкта (наприклад, будівництва), а дуги — роботи, що ведуться на цьому об'єкті.

б) вершини графа відображають роботи, а зв'язки між ними — залежності між роботами.

У методі PERT здебільшого використовують перший варіант представлення графа, у якому дуги відповідають роботам.

Кожній дузі ставиться у відповідність час, за який здійснюється робота і/або кількість ресурсів, що необхідні для виконання роботи.

Основними у мережевих графіках є поняття роботи, події, шляху.

Термін "робота" включає три поняття:

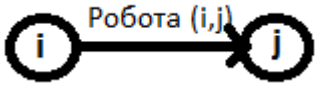
1) фактична робота - тобто трудовий процес, що приводить до досягнення певних результатів і потребує витрат часу і ресурсів;

2) очікування - технологічна перерва в роботі, не вимагає витрат праці(ресурсів), але вимагає витрат часу (наприклад, висихання фарби, твердіння цементу і т.д.);

3) залежність (фіктивна робота) - логічний зв'язок між подіями, що не вимагає витрат часу і ресурсів, але показує, що можливість початку однієї роботи залежить від результатів іншої.

На мережевих графіках фактичну роботу і очікування зображають суцільними стрілками, а залежності - пунктирними.

Подія – це момент часу, коли закінчуються одні роботи і починаються інші. Подія є результатом проведених робіт і, на відміну від робіт, не має тривалості в часі. Роботи зображуються стрілками, які з'єднують вершини якими позначають події. Початок і кінець будь-якої роботи описується парою подій, які називаються початковою і кінцевою подіями. Тому для позначення конкретної роботи використовують код роботи (i,j), який складається з номерів початкової (i-ої) та кінцевої (j-ої) події.



Кожна подія може вважатися такою, що настала, тільки тоді, коли закінчаться усі вхідні в неї роботи. Тому роботи, що виходять із деякої події, не можуть початися, поки не будуть закінчені всі роботи, що входять в цю подію. Подія, яка не має передуючих їй подій, тобто та з якої починається проект, називається вихідною (початковою). Подія, яка не має наступних подій і відображає кінцеву ціль проекту, називається кінцевою. Усі інші події називаються проміжними.

Будь-яка послідовність робіт від однієї події до іншої називається шляхом. Довжина будь-якого шляху визначається сумарною тривалістю робіт, які у нього входять.

Повний шлях - це шлях від вихідної до кінцевої події. У мережевому графіку, як правило, є кілька повних шляхів з різною тривалістю.

Повний шлях, що має максимальну тривалість, називають критичним шляхом. Роботи, які лежать на критичному шляху, називається критичними роботами.

Критичний шлях (L_{kp}) визначає загальну тривалість виконання всього комплексу робіт.

Повні шляхи, тривалість яких менша ніж критичного, називають некритичними. У них є резерв часу, в межах якого час виконання робіт може бути збільшено, що не призводить до збільшення загальної тривалості проекту.

Вихідні дані для побудови мережевого графіку можуть задаватися різними способами, наприклад:

- описом запропонованого проекту; в цьому випадку необхідно самостійно розбити його на окремі роботи і встановити їх взаємозв’язки;

- списком робіт проекту; необхідно проаналізувати зміст робіт і встановити існуючі між ними зв’язки;

- списком робіт проекту із вказаним впорядкуванням; необхідно тільки

відобразити роботи на мережевому графіку.

При побудові мережевого графіка необхідно дотримуватись наступних правил:

- довжина стрілки не залежить від часу виконання роботи;

- стрілка може не бути прямолінійним відрізком;

- для дійсних робіт використовуються неперервні, а для фіктивних пунктирні стрілки;

- кожна операція повинна бути представлена тільки одною стрілкою;

- між одними і тими ж подіями не повинно бути паралельних робіт, тобто робіт з однаковими кодами;

- слід уникати перехрещення стрілок;

- номер початкової події повинен бути меншим за номер кінцевої події;

- не повинно бути подій, які не мають попередників, крім вихідної;

- не повинно бути подій, які не мають наступників, крім кінцевої;

- не повинно бути циклів.

3. Метод критичного шляху

У 1956 році у США М.Уолкер з фірми “Дюпон”, досліджуючи можливості більш ефективного використання обчислювальної машини Univac, об’єднав свої зусилля з Д. Келлі з групи планування капітального будівництва фірми “Ремінгтон Ренд”. Вони спробували використовувати ЕОМ для складання планів-графіків великих комплексів робіт з модернізації заводів фірми “Дюпон”. У результаті був створений раціональний і простий метод опису проекту з використанням ЕОМ. Спочатку він був названий методом Уолкера-Келлі, а пізніше отримав назву Методу Критичного Шляху.

Метод критичного шляху (critical path method, CPM) — це метод планування робіт в рамках проекту, включаючи управління цими роботами і складання графіку їхнього виконання. Ключовим моментом методу є поняття “критичного шляху”.

Критичний шлях — шлях, сумарна тривалість виконання всіх робіт якого є найбільшою.

Мінімальний час, необхідний для виконання будь-якого проекту, дорівнює довжині критичного шляху. Саме на роботи, що належать критичному шляху, слід звертати особливу увагу. Якщо така робота буде відкладена на деякий час, то і термін закінчення проекту буде відкладений на той же час. Якщо необхідно скоротити час виконання проекту, то в першу чергу потрібно скоротити час виконання хоча б однієї роботи на критичному шляху.

Для того, щоб знайти критичний шлях, досить перебрати всі шляхи і вибрати той або ті з них, що мають найбільшу сумарну тривалість виконання робіт. Проте для великих проектів реалізація такого підходу пов’язана з обчислювальними труднощами. Метод CPM дозволяє отримати критичний шлях набагато простіше.

У методі критичного шляху використовуються наступні позначення:

i та j — вершини, або події, проекту;

(i,j) — робота проекту;

s — подія “початок проекту” (start);

f — подія “закінчення проекту” (finish);

T — довжина критичного шляху;

t(i,j)—час виконання роботи (i, j);

ES(i,j)—найбільш ранній час початку роботи (i,j);

EF(i,j)—найбільш ранній час закінчення роботи (i,j);

LS(i,j)—найбільш пізній час початку роботи (i,j)

LF(i,j)—найбільш пізній час закінчення роботи (i,j)

E(i) — найбільш ранній час настання події i;

L(i)— найбільш пізній час настання події i;

R(i,j) — повний резерв часу на виконання роботи (i,j) (час, на який може бути відкладена робота (i,j) без збільшення тривалості виконання всього проекту);

r(i,j) — вільний резерв часу на виконання роботи (i,j) (час, на який може бути відкладена робота (i,j) без збільшення найбільш раннього часу настання наступної події).

Якщо (i,j) — робота проекту, то мають місце співвідношення:

1) для будь-якого j, $ES(i,j)= E(i)$; (1)

2) для будь-якого i, $LF(i,j)= L(j)$. (2)

Для того, щоб використовувати метод CPM для знаходження критичного шляху, необхідно для кожної роботи (i,j) визначити найбільш ранній час початку і закінчення роботи (ES(i,j) та EF(i,j)) і найбільш пізній час початку і закінчення роботи (LS(i,j) та LF(i,j)).

Метод CPM описується наступними співвідношеннями:

$$ES(s,j) = 0 , \quad (3)$$

для будь-якої роботи (s,j), що виходить із стартової вершини s проекту;

$$EF(i,j) = ES(i,j) + t(i,j) = E(i) + t(i,j) , \quad (4)$$

тобто найбільш ранній час закінчення будь-якої роботи (i,j) перевищує найбільш ранній час початку цієї роботи (час настання попередньої події i) на час її виконання;

$$ES(q,j) = \max_i EF(i,q) = E(q), \quad (5)$$

тобто найбільш ранній час початку роботи (q, j) дорівнює найбільшому із значень найбільш раннього часу закінчення безпосередньо передуючих їй робіт;

$$T = \max_i EF(i,f) = E(f), \quad (6)$$

тобто довжина критичного шляху дорівнює найбільш ранньому часу завершення проекту;

$$LF(i,f) = T, \quad (7)$$

тобто найбільш пізній час закінчення будь-якої роботи, що завершує проект, дорівнює довжині критичного шляху;

$$LS(i,j) = LF(i,j) - t(i,j) = L(j) - t(i,j), \quad (8)$$

тобто найбільш пізній час початку будь-якої роботи менше найбільш пізнього часу закінчення цієї роботи (часу настання наступної події) на час її виконання;

$$LF(i,q) = \min_j LS(q,j) = L(q), \quad (9)$$

тобто найбільш пізній час закінчення роботи (i, q) дорівнює найменшому із значень найбільш пізнього часу початку безпосередньо наступних після неї робіт;

$$R(i,j) = LS(i,j) - ES(i,j) = LF(i,j) - EF(i,j) = L(j) - t(i,j) - L(i), \quad (10)$$

тобто повний резерв часу на виконання будь-якої роботи дорівнює різниці між найбільш пізнім і найбільш раннім часом її початку або різниці між найбільш пізнім і найбільш раннім часом її закінчення;

$$r(i,j) = L(j) - ES(i,j) - t(i,j) = L(j) - EF(i,j) = L(j) - E(i) - t(i,j), \quad (11)$$

тобто вільний резерв часу на виконання будь-якої роботи дорівнює різниці між найбільш пізнім часом настання наступної події і найбільш раннім часом закінчення роботи.

З приведених вище визначень і співвідношень безпосередньо витікають наступні твердження:

1. Довжина критичного шляху дорівнює T.

2. Якщо R(i,j) = 0, то робота (i,j) лежить на критичному шляху;

якщо R(i,j) > 0, то робота (i,j) не лежить на критичному шляху.

3. Якщо час початку роботи (i,j), яка не лежить на критичному шляху, відкласти на термін менший, ніж r(i,j), то найбільш ранній час настання наступної події не зміниться.

4. Якщо час початку роботи (i,j), яка не лежить на критичному шляху, відкласти на термін менший, ніж R(i,j), то час, необхідний на виконання всього проекту не збільшиться.

4. Метод PERT

Паралельно з методом критичного шляху у військово-морських силах США був створений метод аналізу й оцінки програм PERT. Даний метод був розроблений корпорацією “Локхід” і консалтинговою фірмою “Буз, Аллен і Гамільтон” для реалізації проекту розробки ракетної системи “Поляріс”. Керівництво програмою виявилося настільки успішним, що проект вдалося завершити на два роки раніше запланованого терміну. Завдяки такому успішному початку даний метод управління незабаром став використовуватися для планування проектів у всіх збройних силах США.

Program (Project) Evaluation and Review Technique (PERT) — техніка оцінки та аналізу програм (проектів), яка використовується при управлінні проектами. PERT — це спосіб аналізу завдань, необхідних для виконання проекту. Особливо, аналізу часу, який потрібен для виконання кожної окремої задачі, а також визначення мінімального необхідного часу для виконання всього проекту.

Метод PERT орієнтований на аналіз таких проектів, для яких тривалість виконання всіх або деяких робіт не вдається визначити точно. Перш за все мова йде про проектування і впровадження нових систем. У таких проектах багато робіт не мають аналогів. В результаті виникає невизначеність у термінах виконання проекту в цілому.

Застосування методу PERT дозволяє отримати відповіді на наступні питання:

1. Чому дорівнює очікуваний час виконання роботи?

2. Чому дорівнює очікуваний час виконання проекту?

3. З якою вірогідністю проект може бути виконаний за вказаний час?

За допомогою методу PERT можна визначити варіацію часу виконання роботи, проекту.

Для того, щоб використовувати метод PERT, для кожної роботи *i*, час виконання якої є випадковою величиною, необхідно визначити наступні три оцінки:

a_i — оптимістичний час (час виконання роботи *i* в найбільш сприятливих умовах);

m_i — найбільш вірогідний (нормальний) час (час виконання роботи *i* в нормальних умовах);

b_i — песимістичний час (час виконання роботи *i* в несприятливих умовах).

Враховуючи, що час виконання роботи добре описується бета-розподілом, середній, або очікуваний, час t_i виконання роботи *i* може бути обрахований за формулою :

$$t_i = (a_i + 4*m_i + b_i) / 6. \qquad (12)$$

Якщо час виконання роботи відомий точно і дорівнює d_i, то $t_i = a_i = m_i = b_i = d_i$.

Маючи в своєму розпорядженні вказані три оцінки часу виконання роботи, можна розрахувати загальноприйняту статистичну міру невизначеності-дисперсію σ^2_i або варіацію var_i часу виконання роботи *i*:

$$\sigma^2_i = var_i = [(b - a) / 6]^2 . \qquad (13)$$

Якщо час виконання роботи *i* відомий точно то $\sigma^2_i = var_i = 0$. (14)

Нехай *T* — час, необхідний для виконання проекту. Якщо в проекті є роботи з невизначеним часом виконання, то час *T* є випадковою величиною.

Математичне очікування (очікуване значення) часу виконання проекту E(T) дорівнює сумі очікуваних значень часу виконання робіт, що лежать на критичному шляху.

Для визначення критичного шляху проекту може бути використаний метод CPM. На цьому етапі аналізу проекту час виконання роботи вважається рівним очікуваному часу t_i.

Варіація (дисперсія) σ^2 (T) загального часу, потрібного для завершення проекту, в припущенні про незалежність часу виконання робіт дорівнює сумі варіацій (дисперсій) часу виконання робіт критичного шляху. Якщо ж дві або більше робіт взаємозалежні, то вказана сума дає наближене уявлення про варіацію часу завершення проекту.

Розподіл часу T завершення проекту є асимптотично нормальним з середнім E(T) і дисперсією σ^2 (T). З урахуванням цього можна розрахувати вірогідність завершення проекту у встановлений термін T_0. Для визначення вірогідності того, що $T \leq T_0$, слід використовувати таблицю розподілу величини

$$z = [T_0 - E(T)]/\sigma(T), \qquad (15)$$

яка має стандартний нормальний розподіл.

Залежність ймовірності виконання проекту за деякий час T від величини Z наведено у Табл. 1.

Z	-2,0	-1,5	-1,0	-0,7	-0,5	-0,3	-0,1	0,1	0,3	0,5	0,7	1,0	1,5	2,0
Імовірність	0,02	0,07	0,16	0,24	0,31	0,36	0,38	0,54	0,62	0,69	0,76	0,84	0,93	0,98

Табл.1 . Розподіл величини Z

5. Управління проектами в галузі розробки програмного забезпечення

Динамічний розвиток і взаємопроникнення бізнес-технології призводить до значних якісних змін ролі та функції інформаційних технологій на сучасному підприємстві. Найкращим інструментом для запровадження інформаційних технологій в ринкових умовах є ІТ-проект (проект інформатизації). Такі проекти характеризуються абстрактністю продукту, нелінійністю процесу розробки, збільшенням ризиків в кінці життєвого циклу тощо.

ІТ - проект — це комплекс формально організованих заходів з метою досягнення єдиної мети, створення складної системи із встановленими характеристиками якості та обмежених ресурсах. Такого роду проект являє собою набір взаємно пов’язаних ресурсів, що забезпечує випуск одного чи декількох ІТ-продуктів, для клієнта чи кінцевого користувача. Зазвичай цей набір визначається на початку проекту та керується згідно зі встановленим планом. На практиці такі проекти являють собою сукупність процесів, що забезпечують зміни технологічних чи соціальних систем.

В Законі України «Про Концепцію Національної програми інформатизації» під проектом інформатизації розуміється комплекс взаємопов’язаних заходів, зазвичай інвестиційного характеру, що узгоджені за часом, використанням певних матеріально-технічних, інформаційних, людських, фінансових та інших ресурсів і мають на меті створення заздалегідь визначених інформаційних і телекомунікаційних систем, засобів інформатизації та інформаційних ресурсів, які відповідають певним технічним умовам і показникам якості. Узагальнюючи викладене, під проектом інформатизації (ІТ-проектом) можна розуміти сукупність процесів та комплексу дій, пов’язаних зі створенням, удосконаленням чи впровадженням інформаційної системи або її складової частини — програмного забезпечення.

Сучасний ріст ІТ-ринку характеризується структурними зрушеннями в сторону складних ІТ-проектів, цілі яких сформульовані вимогами бізнесу.

Розробка програмного забезпечення включає в себе багато стадій: проектування, програмування, тестування, впровадження і підтримку.

Проектування починається із формулювання вимог до програмного забезпечення і створення специфікацій — документів, у яких описані функції, що їх повинна виконувати програма. На наступному етапі створюється загальний дизайн програми: розбиття її на окремі блоки і визначення взаємодії між ними. На етапі безпосереднього програмування створюється текстовий код програми на одній чи декількох мовах програмування. Після компіляції коду, програмний продукт обов'язково проходить тестування, у процесі якого визначається відповідність продукту специфікаціям, знаходяться і виправляються помилки.

Перед впровадженням програмний продукт потребує документації — опису можливостей, посібників користувача, системи допомоги. Після впровадження програмного забезпечення, що для програмних продуктів вимагає маркетингу, системи дистрибуції, реклами тощо, програмне забезпечення потребує підтримки. Необхідність у підтримці виникає внаслідок швидкого розвитку комп'ютерів, що зумовлює необхідність взаємодії програмного продукту з іншими, новішими програмами і новою матеріальною базою. Часто підтримка нових можливостей забезпечується випуском нових версій програмного продукту.

6. Середовище розробки Microsoft Visual Studio

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Середовище розробки Visual Studio може працювати і компілювати проекти на багатьох мовах: Visual C++, Visual C#, Visual Basic і т.д. Багато варіантів постачання також включають Microsoft SQL Server або MSDE Visual Source Safe — файл-серверні системи управління версіями.

Visual Studio 2010. Представлений 12 квітня 2010 року. Включає .NET Framework 4.0. З'явилася нова мова F#, Visual C++ підтримує стандарт C++0x. Інструменти Visual Studio 2010 допоможуть не тільки в створенні звичних програм для мобільних телефонів і персональних комп'ютерів, але в розробці хмарних застосунків.

У Visual Studio 2010 повністю перероблений інтерфейс з використанням Windows Presentation Foundation (WPF), упроваджено наступне покоління інструментів ASP.NET, є підтримка динамічних розширень в мовах програмування C# і Visual Basic, використовуються нові шаблони проектів, інструментарій для документування тестових сценаріїв і велика кількість нових бібліотек, що підтримують Windows 7.

Також включає Visual Studio Team System 2010 - інструмент для спільної розробки застосунків.

7. Мова програмування C#

C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

C# на сьогоднішній день є однією із найпотужніших об’єктно-орієнтовних мов програмування. Крім того, ця мова повністю орієнтується на платформу .NET Framework, що забезпечує ефективне використання максимальної кількості можливостей останньої.

Мова C# також досить зручна в програмуванні, що в поєднанні з інструментальними засобами середовища MS Visual Studio, робить процес розробки максимально «приємним» для програміста.

C# – відносно нова мова програмування, що характеризується наступними основними можливостями:

– повна підтримка об’єктно-орієнтованого програмування;

– чітко визначений набір базових типів;

– вбудована підтримка автоматичної генерації XML-документації;

– автоматичне звільнення виділеної динамічної пам'яті;

– можливості маркування властивостей і методів атрибутами;

– повна підтримка бібліотеки класів .NET;

– можливість прямого доступу до пам’яті через вказівники;

– підтримка властивостей та подій;

– можливості використання для написання динамічних сторінок ASP.NET та Web-служб XML.

Варто зауважити, що хоча й C# призначена для розробки коду, що виконується середовищем .NET, сама вона не є частиною .NET. Існують деякі можливості, які підтримуються .NET, але не підтримуються C#. І, як не дивно, існують можливості C#, які не підтримуються .NET.

8. Робота з базами даних

8.1. База даних

База даних (БД) — впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно, та призначені для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД.

Головним завданням БД є гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин: збереженої інформації та системи управління нею.

8.2. Система управління базами даних Access

Системою управління базами даних (СУБД, DBMS - Data Base Management System) називають програму, призначену для створення та ведення баз даних, а також організації доступу до даних і їх обробки.

База даних організовується відповідно до моделі даних, яка підтримується в СУБД. Реляційна модель даних (англ. Relation - відношення) є однією з найпоширеніших моделей, що використовуються в сучасних СУБД. Реляційна модель орієнтована на організацію даних у вигляді прямокутних двомірних таблиць.

СУБД Access відноситься до СУБД реляційного типу, що працює в середовищі Windows. Цей програмний продукт є складовою частиною Microsoft Office.

Об'єкти Access:

Таблиці-складають основу бази даних і призначені для зберігання інформації про об'єкти предметної області.

Запити - є засобом вибірки необхідних даних з однієї або декількох таблиць БД.

Форми-являють собою електронний варіант фізичних форм документів. Вони призначені для введення, перегляду і редагування даних.

Звіти-використовуються для формування вихідних документів, призначених для виводу на екран, принтер або у файл.

Макроси-містять опис дій, які повинні бути виконані у відповідь на деяку подію. Кожна дія реалізується макрокомандою.

Модулі-містять програми мовою Visual Basic, які розробляються користувачем для реалізації нестандартних процедур обробки даних в задачах користувача.

Для створення об'єктів бази даних (таблиць, запитів, форм, звітів) використовуються спеціалізовані діалогові графічні засоби: Конструктор (Design), а також програми-майстри Access (Wizard). Поряд з діалоговими засобами створення об'єктів БД, Access підтримує такі засоби програмування:

· SQL (Structured Query Language) - структурована мова запитів;

· Мову макрокоманд;

· VBA (Visual Basic for Applications) - об'єктно-орієнтована мова програмування.

8.3. SQL-запити

SQL (Structured query language — мова структурованих запитів) — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікації, система контролю за доступом до бази даних.. Сам по собі SQL не є ні системою керування базами даних, ні окремим програмним продуктом.

SQL – це діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також управління базами даних. Багато баз даних підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, обновлення, і вилучення даних, використовуючи систему управління і адміністративні функції. SQL також включає CLI (Call Level Interface) для доступу і управління базами даних дистанційно.

Перша версія SQL була розроблена на початку 1970-х років у IBM. Ця версія носила назву SEQUEL і була призначена для обробки і пошуку даних, що містилися в реляційній базі даних IBM, System R . Мова SQL пізніше була стандартизована Американськими Держстандартами (ANSI) в 1986. Спочатку

SQL розроблялась як мова запитів і управління даними, пізніші модифікації SQL створено продавцями системи управління базами даних, які додали процедурні конструкції, control-of-flow команд і розширення мов. З випуском стандарту SQL:1999 такі розширення були формально запозичені як частина мови SQL через Persistent Stored Modules (SQL/PSM).

Загалом мову SQL поділяють на дві частини: мову маніпуляції даними (Data Manipulation Language (DML)), та мову опису даних (Data Definition Language (DDL)).

Мова SQL не чутлива до регістру, і оператори пишуть великими буквами тільки для зручності.

Мова опису даних

Дозволяє нам описати структуру даних.

Нехай маємо базу даних університету, і користувача, що має повний доступ до цієї бази. Створимо таблицю з людьми:

```
CREATE TABLE persons (
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  surname VARCHAR(50),
  name VARCHAR(200),
  tel VARCHAR(50)
);
```

Щоб потім змінити структуру таблиці використовують команду ALTER, наприклад:

```
ALTER TABLE persons ADD COLUMN email VARCHAR(100);
```

Крім цього таблиці можна перейменовувати

```
ALTER TABLE TABLE RENAME AS new_table;
```

та звісно видаляти поля:

```
ALTER TABLE persons DROP COLUMN tel;
```

Мова маніпуляції даними

Є чотири основних типи запитів даних в SQL, які належать до мови маніпулювання даними:

SELECT - вибрати рядки з таблиць;

INSERT - додати рядки в таблицю;

UPDATE - змінити рядки в таблиці;

DELETE - видалити рядки в таблиці;

SELECT

Оператор SELECT дозволяє вибирати дані з бази. Загалом оператор SELECT виглядає так:

SELECT імена_полів

FROM імена_таблиць

WHERE умова;

Імена полів записуються через кому. Якщо потрібно вибрати всі поля, пишуть зірочку ("*"). При потребі можна уточнити з якої таблиці брати поле, додавши ім’я таблиці перед іменем поля.

SELECT може видати нам рядки, що повторюються. Якщо ми хочемо мати тільки унікальні значення, то можемо уникнути повторень командою DISTINCT.

SELECT DISTINCT поля FROM таблиці;

Умова дозволяє відкинути непотрібні нам значення. Загалом в умові певні поля порівнюються з певними значеннями, чи між собою. Текстові значення беруться в одинарні лапки (можна і в подвійні). Для порівняння можна користуватись такими операторами:

Оператор	*Опис*
=	Рівність
<> (можливо також !=)	Нерівність
<, >, <=, >=	Менше, більше, менше рівне, більше рівне
BETWEEN	Всі значення що знаходяться між двома даними включно

LIKE	Порівняння з шаблоном
IN	Приймає одне з перелічених значень

Табл. 2. Оператори SQL-запиту SELECT

Також в умові можна використовувати оператори OR, AND та NOT, та дужки.Також до запиту SELECT можна додати команду ORDER BY, що дозволяє впорядкувати результат за заданими стовпцями. Щоб сортувати в зворотньому порядку після стовпців за якими сортують пишуть DESC:

SELECT name FROM students ORDER BY name DESC;

Також можна задати максимальну кількість записів в результаті. Корисно для великих таблиць. Правда синтаксис трохи відрізняється для різних систем.

INSERT

Оператор INSERT додає до таблиці рядок. Має такий синтаксис:

INSERT INTO назва_таблиці VALUES (список_значень);

Значення мають йти в такому ж порядку, як і стопці таблиці.

UPDATE

Змінює значення полів в уже існуючих записах. Синтаксис:

UPDATE назва_таблиці SET стовпець1=значення1, стовпець2=значення2, ... WHERE умова;

DELETE

Видаляє рядки таблиці. Синтаксис :

DELETE FROM назва_таблиці WHERE умова .

8.4. Простір імен System.Data.OleDb

Для організації взаємодії програми з базою даних використовується простір імен System.Data.OleDb .

System.Data.OleDb є постачальником даних. NET Framework для OLE DB і являє собою набір класів, що використовуються для доступу до джерел даних OLE DB в керованому просторі.

Клас	Опис
OleDbCommand	Представляє оператор SQL або збережену процедуру, що застосовується до джерела даних.
OleDbCommandBuilder	Автоматично генерує однотабличні команди, які дозволяють узгодити зміни, що вносяться в об'єкт DataSet, зі зв'язаною базою даних. Цей клас не успадковується.
OleDbConnection	Надає відкрите підключення до джерела даних.
OleDbConnectionString Builder	Забезпечує зручний спосіб створення й керування вмістом рядків підключення за допомогою класу OleDbConnection.
OleDbDataAdapter	Представляє набір команд даних і підключення бази даних, які використовуються для заповнення DataSet і оновлення джерела даних.
OleDbDataReader	Надає спосіб читання потоку рядків даних з джерела тільки в прямому порядку. Цей клас не успадковується.
OleDbEnumerator	Надає механізм для перерахування всіх доступних постачальників OLE DB в локальній мережі.

OleDbError	Збирає відомості, що мають відношення до попередження або помилки, поверненої джерелом даних.
OleDbErrorCollection	Збирає всі помилки, згенеровані постачальником даних .NET Framework для OLE DB. Цей клас не успадковується.
OleDbException	Виключення, яке генерується, коли відповідний постачальник повертає попередження або помилку для джерела даних OLE DB. Цей клас не успадковується.
OleDbFactory	Представляє набір методів для створення екземплярів класів постачальників OLE DB, що реалізують джерело даних.
OleDbInfoMessageEvent Args	Надає дані для події InfoMessage. Цей клас не успадковується.
OleDbMetaDataCollectionNames	Надає список констант для використання з методом GetSchema з метою отримання колекцій метаданих.
OleDbMetaDataColumn Names	Надає статичні значення, які використовуються для імен стовпців в об'єктах OleDbMetaDataCollection Names, що містяться в таблиці DataTable. Об'єкт DataTable створений за допомогою методу GetSchema.

OleDbParameter	Представляє параметр для OleDbCommand і додатково його відображення для стовпця DataSet. Цей клас не успадковується.
OleDbParameterCollection	Представляє колекцію параметрів, які важливі для OleDbCommand, а також їх співставлення із стовпцями в DataSet.
OleDbPermission	Дозволяє постачальнику даних. NET Framework для OLE DB забезпечити користувачеві рівень безпеки, достатній для доступу до джерела даних OLE DB.
OleDbPermissionAttribute	Пов'язує операцію безпеки з користувацьким атрибутом безпеки.
OleDbRowUpdatedEvent Args	Надає дані для події RowUpdated.
OleDbRowUpdatingEventArgs	Надає дані для події RowUpdating.
OleDbSchemaGuid	Повертає тип таблиці схеми, що вказується методом GetOleDbSchemaTable.
OleDbTransaction	Представляє створювану в джерелі даних SQL-транзакцію. Цей клас не успадковується.

Табл. 3. Класи простору імен System.Data.OleDb

9. Програмна реалізація повнофункціонального методу PERT

Було створено програму, яка повністю реалізує функціональність методу PERT управління проектами. Програму написано мовою C# у середовищі Microsoft Visual Studio 2010. Повний код програми наведений у додатку "Код програми".

9.1. Інтерфейс і основні функції програми

Програма має зручний графічний інтерфейс для менеджерів проектів.

На формі розміщено компоненту tabControl - це панель вкладок, з допомогою якої можна зекономити місце на формі. TabControl має наступні вкладки: "Список робіт", "Порахувати ймовірність вчасного виконання", "Додати роботу", "Заповнити список робіт за допомогою матриць", "Налаштування". Детальний опис вкладок наведено нижче.

Рис. 3. Вкладка "Список робіт"

На Рис. 3 показано вигляд вкладки "Список робіт". На цій вкладці розміщено компоненту dataGridView1, що представляє собою таблицю, у яку заносяться характеристики робіт проекту. Таблиця включає у себе наступні стовпці, у які записуються відповідні характеристики роботи :

"ID Початку операції", "ID Кінця операції", "Оптимістична оцінка тривалості", "Найбільш ймовірна оцінка тривалості", "Песимістична оцінка тривалості" – ці характеристики є вихідними даними, які вносить користувач;

"Середньозважений час операції", "Середнє відхилення тривалості

операції”, “Ранній старт”, “Ранній фініш”, “Пізній старт”, “Пізній старт”, “Пізній фініш”, “Загальний часовий резерв”, “Належність до критичного шляху” – ці параметри обраховуються програмою за допомогою методів PERT і CPM.

Обчислення названих вище параметрів відбувається при натисканні на кнопку “Обчислити”. Також обчислюються критичний шлях та загальне відхилення тривалості проекту, значення яких виводяться у відповідних компонентах textBox під списком робіт проекту. Критичний шлях проекту рівний його прогнозованій тривалості.

Обчислення відбуваються за допомогою наступних функцій :

```
for (int i = 0; i < dataGridView1.RowCount - 1; i++)
{
Act[i].duration = (optKoef * Act[i].optimistic + mostKoef * Act[i].mostlikely + pessKoef * Act[i].pessimistic) / totalKoef;
Act[i].sigma = (pessKoef * Act[i].pessimistic - optKoef * Act[i].optimistic) / totalKoef;
}

public void Critical_Path_Method(acts[] Act)
    {
      for (int i = 0; i < dataGridView1.RowCount - 1; i++)
      {
        Act[i].earlyStart = 0;
        Act[i].earlyFinish= 0;
        for (int j = 0; j < dataGridView1.RowCount - 1; j++)
        {
          if (Act[j].finish == Act[i].start && Act[j].earlyFinish>Act[i].earlyStart)
          {
            Act[i].earlyStart = Act[j].earlyFinish;
```

```
            }
        }
        Act[i].earlyFinish = Act[i].earlyStart + Act[i].duration;
        dataGridView1[7, i].Value = Act[i].earlyStart;
        dataGridView1[8, i].Value = Act[i].earlyFinish;
    }
    for (int i = dataGridView1.RowCount - 2; i >= 0; i--)
    {
        Act[i].lateStart = 0;
        Act[i].lateFinish = 0;
        for (int j = dataGridView1.RowCount - 2; j >= 0 ; j--)
        {
            if (Act[j].finish == Act[i].finish && Act[i].finish == maxV &&
Act[j].earlyFinish > Act[i].lateFinish)
            {
                Act[i].lateFinish = Act[j].earlyFinish;

            }
            else
             if (Act[i].lateFinish ==0)   Act[i].lateFinish = Int32.MaxValue;
            if (Act[j].start == Act[i].finish && Act[j].lateStart < Act[i].lateFinish)
            {
                Act[i].lateFinish = Act[j].lateStart;
            }

        }
        Act[i].lateStart = Act[i].lateFinish - Act[i].duration;
        if (Act[i].lateStart < 0) Act[i].lateStart = 0;
        dataGridView1[9, i].Value = Act[i].lateStart;
        dataGridView1[10, i].Value = Act[i].lateFinish;
```

```
        }
        critWay = 0;
        totalSigma = 0;
        for (int i = 0; i < dataGridView1.RowCount - 1; i++)
        {
           Act[i].fullTimeReserve = Act[i].lateFinish - Act[i].earlyFinish;
           if (Act[i].fullTimeReserve < 0) Act[i].fullTimeReserve = 0;
           if (Act[i].fullTimeReserve == 0)
           {
              Act[i].isCritical = true;
              critWay += Act[i].duration;
              totalSigma += Math.Pow(Act[i].sigma,2);
              dataGridView1[12, i].Style.ForeColor = Color.Red;
              dataGridView1[12, i].Style.Font = new Font(dataGridView1.Font,
FontStyle.Bold);
           }
           dataGridView1[11, i].Value = Act[i].fullTimeReserve;
           dataGridView1[12, i].Value = Act[i].isCritical;
        }
        totalSigma = Math.Sqrt(totalSigma);
        textBox1.Text = critWay.ToString();
        textBox2.Text = totalSigma.ToString();
     }
```

При натисканні кнопки “Очистити список робіт” відповідно відбувається очищення списку робіт проекту. Код відповідної процедури наведено в Додатку.

Кнопка “Зберегти у базі даних” дозволяє зберегти результати роботи програми (а саме, список робіт з обчисленими їх характеристиками та список ймовірностей виконання проекту за деякі проміжки часу) у базі даних Microsoft Access. Для організації взаємодії програми з базою даних використовується

простір імен System.Data.OleDb та SQL - запити. Для перенесення даних у базу використовуються запити INSERT та DELETE.

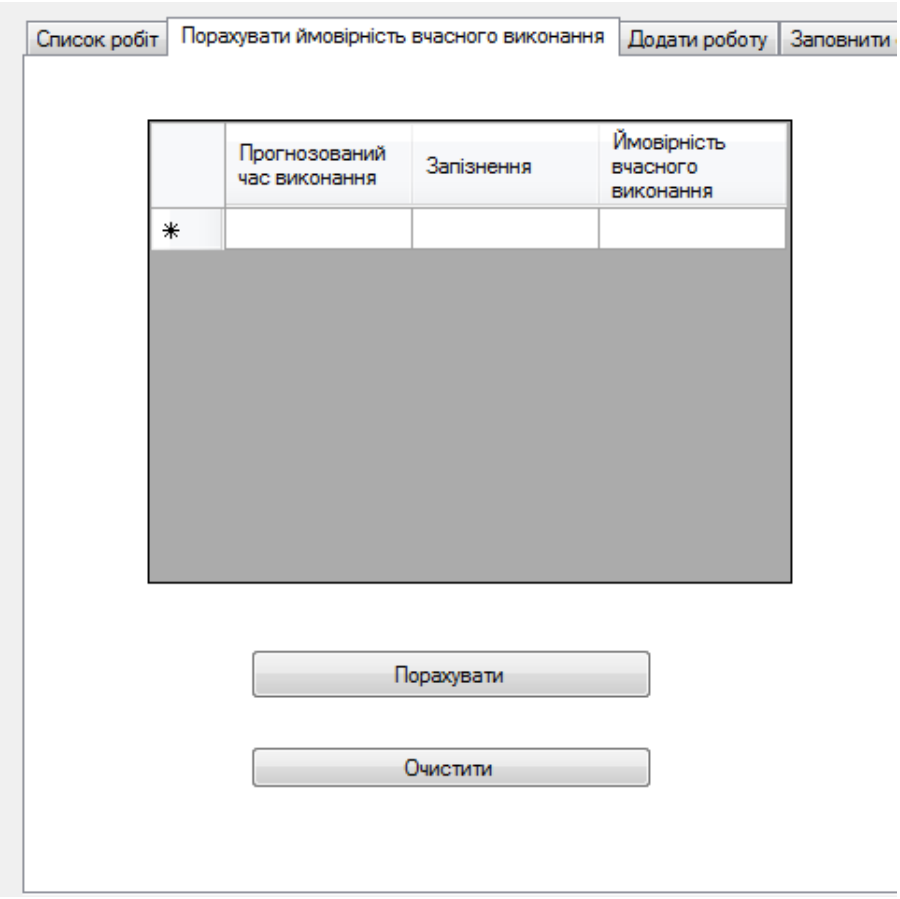


Рис. 4. Вкладка “Порахувати ймовірність вчасного виконання”

На Рис. 4 зображено вкладку “Порахувати ймовірність вчасного виконання”. Вкладка містить компонент dataGridView2 – таблицю з колонками “Прогнозований час виконання”, “Запізнення”, “Ймовірість вчасного виконання”. Також на вкладці розміщено дві кнопки : “Порахувати” та “Очистити”. Кнопка “Очистити” виконує очищення таблиці.

Користувачу потрібно ввести “Прогнозований час виконання”. Одночасно можна обраховувати ймовірності для необмеженої кількості моментів часу. Після натискання кнопки “Порахувати” програма обчислить для кожного з моментів часу запізнення Z, і згідно з таблицею розподілу величини Z, яка наведена у розділі 4 (Табл. 1), визначить ймовірності виконання проекту за заданий час.

У програмному коді цей функціонал реалізовано наступним чином :

```
public void SearchProbability()
{
   double z, p,ts;
   p = 0;
   for (int i = 0; i < dataGridView2.RowCount - 1; i++)
   {
      ts = Convert.ToDouble(dataGridView2[0, i].Value);
      z = (ts - critWay) / totalSigma;
      if (z < -1.5) p = 0.02;
      if (z >= -1.5 && z<-1) p = 0.07;
      if (z >= -1 && z < -0.7) p = 0.16;
      if (z >= -0.7 && z < -0.5) p = 0.24;
      if (z >= -0.5 && z < -0.3) p = 0.31;
      if (z >= -0.3 && z < -0.1) p = 0.36;
      if (z >= -0.1 && z < 0.1) p = 0.38;
      if (z >= 0.1 && z < 0.3) p = 0.54;
      if (z >= 0.3 && z < 0.5) p = 0.62;
      if (z >= 0.5 && z < 0.7) p = 0.69;
      if (z >= 0.7 && z < 1.0) p = 0.76;
      if (z >= 1.0 && z < 1.5) p = 0.84;
      if (z >= 1.5 && z < 2) p = 0.93;
      if (z >= 2) p = 0.98;
      dataGridView2[1, i].Value = z;
      dataGridView2[2, i].Value = p;
   }
}
```

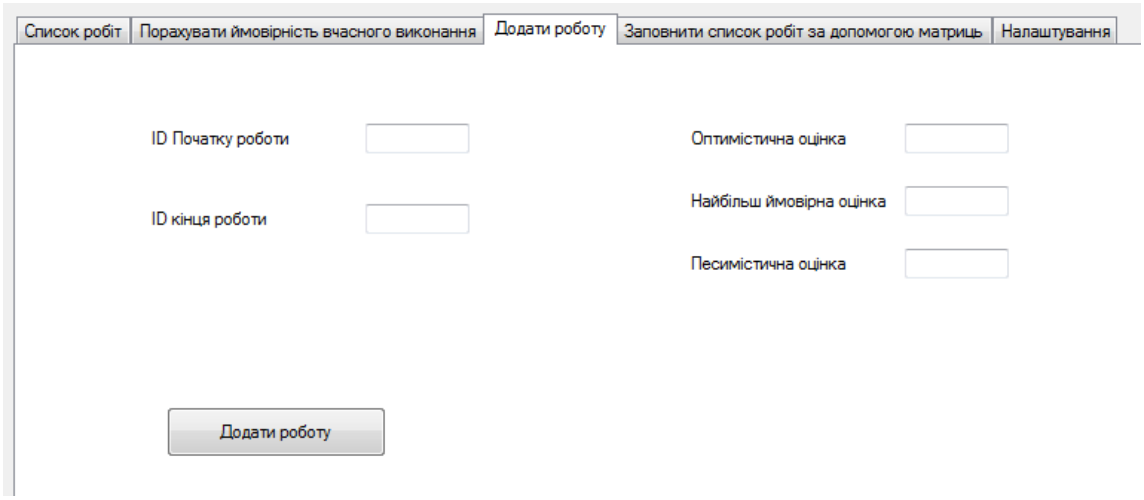


Рис. 5. Вкладка “Додати роботу”

Список робіт можна заповнювати не вручну, а за допомогою вкладки “Додати роботу”, яка зображена на Рис.5. Користувач має ввести необхідні параметри роботи, і після цього натиснути кнопку “Додати роботу”. Після цього буде виведено повідомлення про те, що роботу додано до списку.

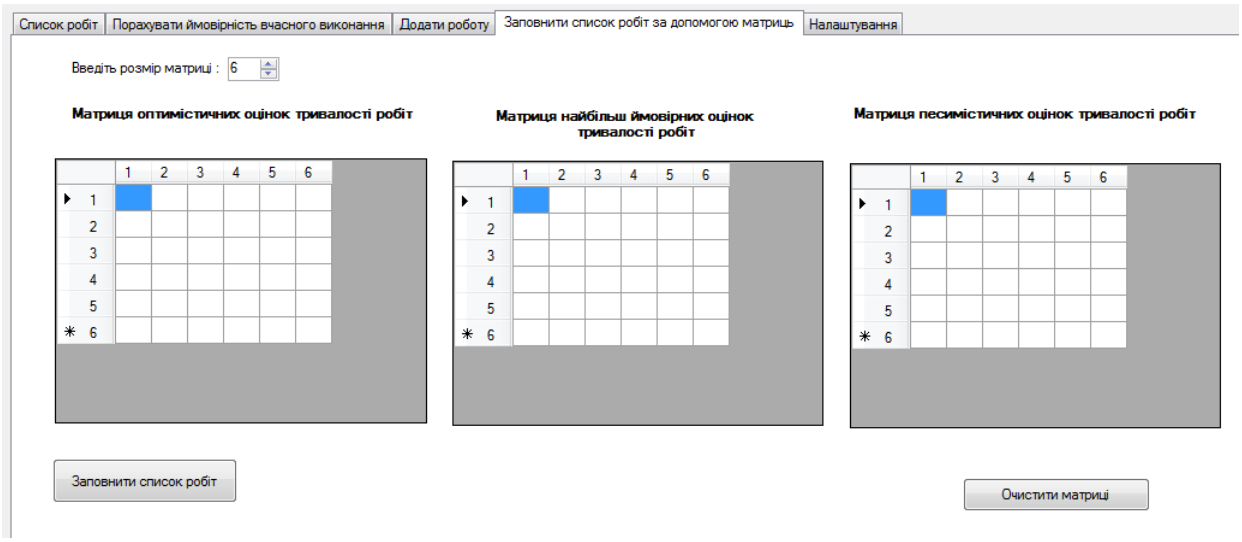


Рис. 6. Вкладка “Заповнити список робіт за допомогою матриць”

Також список робіт можна заповнювати за допомогою матриць суміжності мережевого графа (Рис. 6). Користувач має ввести розмір матриць, який відповідає кількості подій проекту, та занести у матриці вхідні дані. Кнопки “Заповнити список робіт” та “Очистити матриці” відповідно

заповнюють список робіт проекту та очищають матриці суміжності.

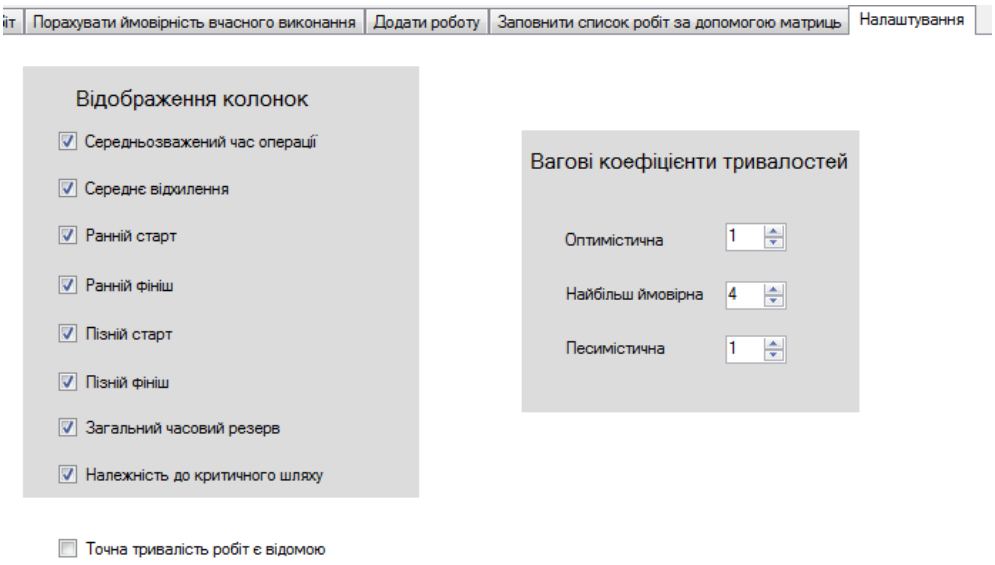


Рис. 7. Вкладка “Налаштування”

Останньою є вкладка “Налаштування”, яку зображено на Рис. 7. Використовуючи цю вкладку можна змінювати вигляд списку робіт, визначаючи які саме стовпці потрібно відображати. Можна змінювати вагові коефіцієнти оцінок тривалості робіт, якщо цього вимагають умови виконання проекту. Також можна вибрати опцію “Точна тривалість робіт є відомою”. Коли ми знаємо точну тривалість робіт, то використання методу PERT непотрібне, і виконується лише метод CPM для обчислення відповідних параметрів проекту.

9.2. Апробація результатів роботи програми

Після створення програми було проведено перевірку правильності її роботи і коректності отриманих результатів.

Для тестування програми було задано мережевий графік проекту, точна тривалість виконання робіт якого невідома, і який включає у себе шість робіт, що характеризуються оптимістичною, найбільш ймовірною та песимістичною оцінками тривалості. Мережевий графік проекту зображено на Рис. 8. Вхідні дані проекту наведено на Рис. 9. Використовуючи форму "Додати роботу", заносимо вхідні дані у програму (Рис. 10).

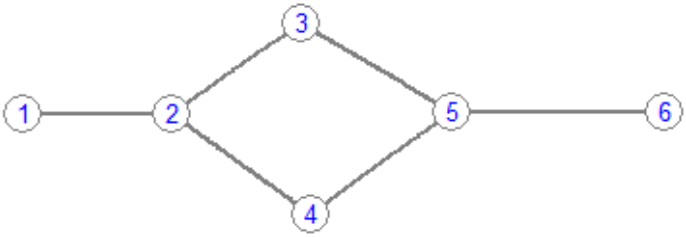


Рис. 8. Мережевий граф тестового проекту

ID Початку операції	ID Кінця операції	Оптимістична оцінка тривалості	Найбільш ймовірна оцінка тривалості	Песимістична оцінка тривалості
1	2	17	29	47
2	3	6	12	24
2	4	16	19	28
3	5	13	16	19
4	5	2	5	14
5	6	2	5	8

Рис. 9. Вхідні дані тестового проекту

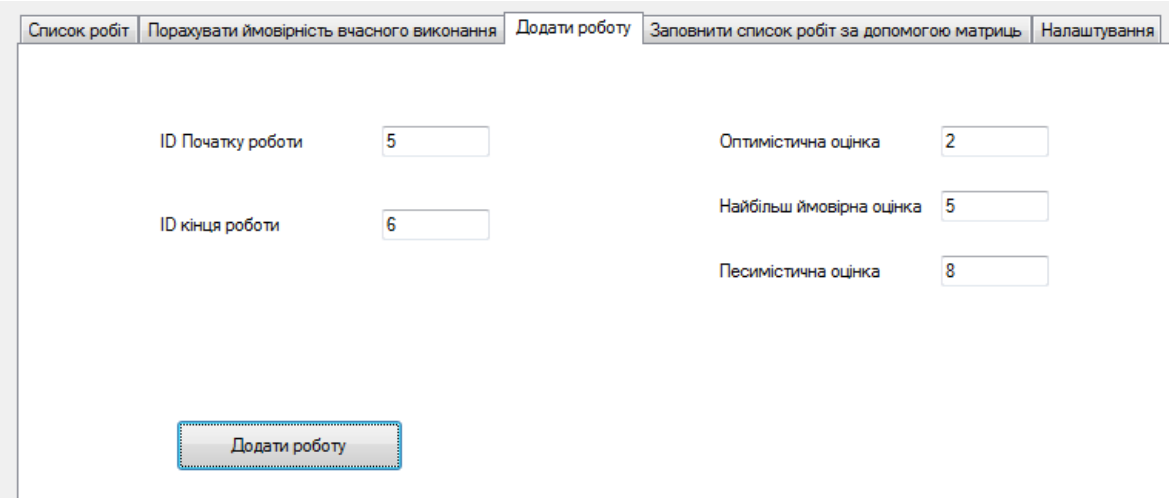
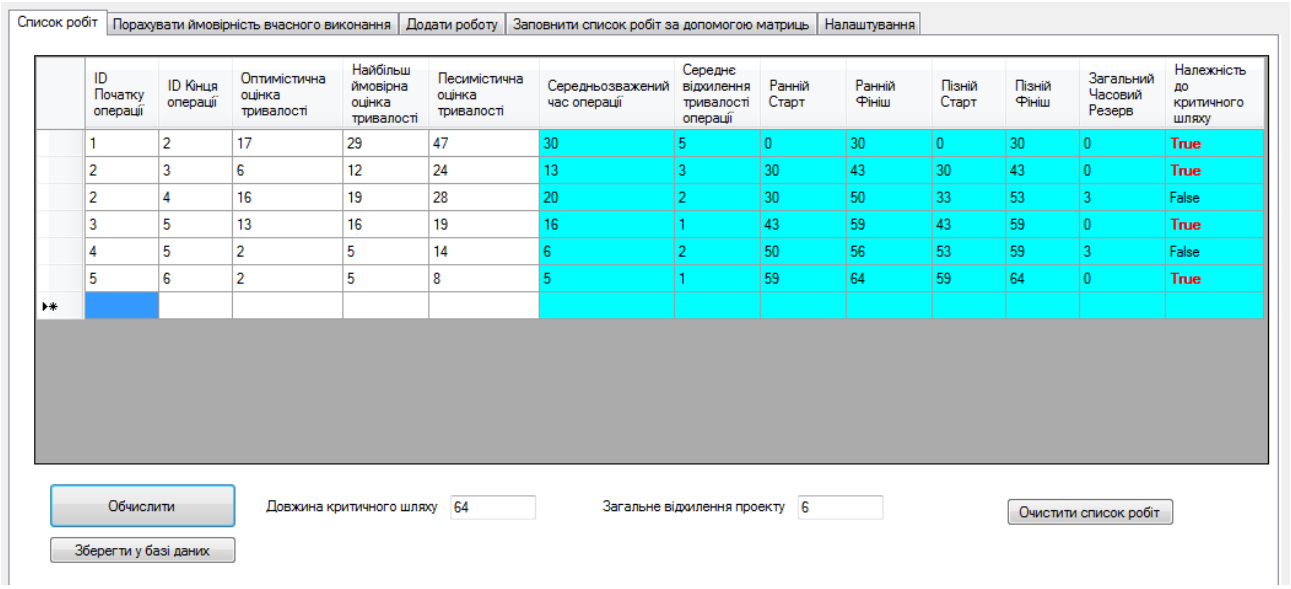


Рис. 10. Додавання робіт у список за допомогою форми “Додати роботу”

Після введення вхідних даних обраховуємо приблизну тривалість та відхилення кожної роботи, прогнозовану тривалість (критичний шлях) та загальне відхилення проекту, та решту характеристик робіт. Результати наведені на Рис. 11.



ID Початку операції	ID Кінця операції	Оптимістична оцінка тривалості	Найбільш ймовірна оцінка тривалості	Песимістична оцінка тривалості	Середньозважений час операції	Середнє відхилення тривалості операції	Ранній Старт	Ранній Фініш	Пізній Старт	Пізній Фініш	Загальний Часовий Резерв	Належність до критичного шляху
1	2	17	29	47	30	5	0	30	0	30	0	True
2	3	6	12	24	13	3	30	43	30	43	0	True
2	4	16	19	28	20	2	30	50	33	53	3	False
3	5	13	16	19	16	1	43	59	43	59	0	True
4	5	2	5	14	6	2	50	56	53	59	3	False
5	6	2	5	8	5	1	59	64	59	64	0	True

Рис.11. Результати обчислень

Далі, використовуючи вкладку “Порахувати ймовірність вчасного виконання”, виконуємо обчислення ймовірності виконання проекту за деякий час. Для прикладу введено дві тривалості, і для кожної з них пораховано ймовірність завершення проекту за даний час. Результати наведено на Рис. 12.

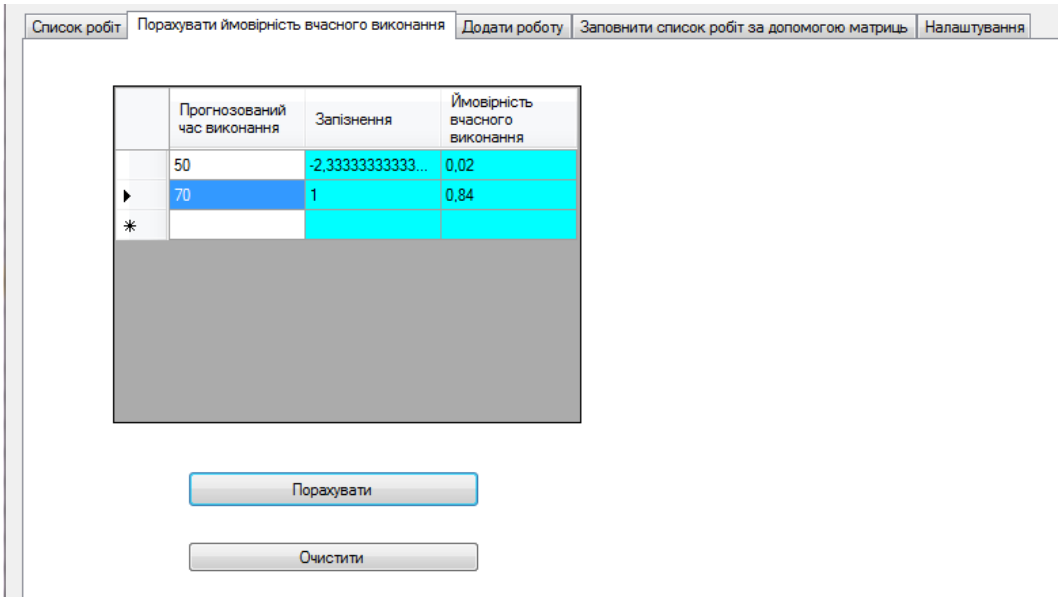


Рис.12. Результати обчислення ймовірностей виконання проекту за заданий час

Зберігаємо результати роботи програми у базі даних Access. Результати збереження наведено на Рис. 13.

СписокРобіт

IdПочатку	IdКінця	Оптимістична	Найбільшим	Песимістичн	Тривалість	СереднєВідх	РаннійСта	ПізнійС	РаннійФі	ПізнійФ	РезервЧасу	Критичний
1	2	17	29	47	30	5	0	30	0	30	0	True
2	3	6	12	24	13	3	30	43	30	43	0	True
2	4	16	19	28	20	2	30	50	33	53	3	False
3	5	13	16	19	16	1	43	59	43	59	0	True
4	5	2	5	14	6	2	50	56	53	59	3	False
5	6	2	5	8	5	1	59	64	59	64	0	True

СписокРобіт | ЙмовірністьВиконання

ПрогнозованийЧас	Запізнення	ЙмовірністьВиконання
50	-2,3333333333	0,02
70	1	0,84

Рис. 13. Результати роботи програми, збережені у базі даних

10. Охорона праці та безпека в надзвичайних ситуаціях

10.1. Вступ

З розвитком науково-технічного прогресу важливу роль грає можливість безпечного виконання людьми своїх трудових обов'язків. Охорона здоров'я людей що працюють, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму становить основу людської безпеки. Проте не варто забувати, що небезпечні чинники можуть діяти на людський організм не-поодинці, а взаємопов'язано. Але нажаль ми не можемо проаналізувати всі можливі комбінації сукупної дії небезпечних та шкідливих чинників. Невід'ємним елементом сьогодення є те, що людина проводить половину свого дня у приміщені за роботою. Мета роботи – вивчення безпеки праці на робочому місці, вплив шкідливих чинників на працюючу особу й відчуття міри захисту від неї. У зв'язку з цим була створена та розвивається наука про безпеку праці та життєдіяльності людини. Це комплекс заходів, вкладених у забезпечення безпеки людини у середовище проживання, збереження здоров'я, розробку методів і засобів захисту за методом зниження впливу шкідливих і найнебезпечніших чинників до допустимих значень, вироблення заходів для обмеження шкоди та ліквідацію наслідків надзвичайних ситуацій мирного й військової часу. Під час виконання бакалаврської роботи було використано ряд різноманітного приладдя яке дає змогу опрацьовувати низку матеріалу, а саме комп'ютери, принтери, сканери. Не враховуючи приміщення в якому відбувається цей тривалий процес, а саме лабораторії, комп'ютерні класи За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі мікрокліматичні умови, освітлення, електромагнітні випромінювання, забруднення повітря шкідливими речовинами (джерелом яких може бути принтер, сканер), шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу .

10.2. Аналіз стану умов праці

10.2.1. Характеристика виробничого середовища та його чинників

Практична частина роботи над дипломним проектом виконувалась в міжкафедральній обчислювальній лабораторії факультету електроніки Львівського національного університету ім. І. Франка. Санітарно-гігієнічні умови в лабораторії відповідають існуючим нормам, а саме:

- температура повітря в холодну пору року знаходиться в межах 18-20 °С при швидкості руху повітря 0.2 м/с.
- в теплу пору року температура в лабораторії не перевищує температури зовнішнього повітря більш ніж на 3-5 °С при швидкості руху повітря 0.3 м/с.
- відносна вологість повітря складає 40-60%.
- покриття підлоги сухе (паркет), тому не проводить електричний струм.
- освітлення приміщення 150-300 лк, тому відповідає нормам.
- обмін повітря в лабораторії здійснюється вентиляцією(в межах 30 м3/год на одного працівника).
- є аптечка з усіма необхідними медикаментами для надання першої медичної допомоги.

10.2.2.Опис процесу праці

При використанні ПЕОМ можливе ушкодження деяких функціональних частин тіла. Наприклад:

- Порушення центральної нервової системи та зорового аналізатора.

Погіршення функціонального стану центральної нервової системи викликають зорове та нервово-емоційне напруження, а також висока відповідальність за покладені завдання. Всі порушення центральної нервової системи впливають на продуктивність праці. Як відомо саме зоровий аналізатор людини несе на собі основне навантаження при роботі користувача комп’ютера. У зв’язку з цим частота порушень зору у користувачів комп’ютерів на 10-20 % більша, ніж серед інших службовців.

Основними факторами впливу є: недостання чіткість символів, високий контраст зображення, недостатня частота розгортки.

- Порушення скелетно-м’язової системи.

Діяльність користувача комп’ютера характеризується тривалою, багатогодинною роботою у сидячому стані, так недотримання відповідних умов може згубно впливати на скелетно-м’язову систему.

- Ураження електричним струмом.

Оскільки комп’ютер живиться від електромережі, то необхідним є дотримання всіх вимог техніки безпеки.

Гранично допустимий струм, що може пройти через людину при нормальному режимі електропристроїв не повинен перебільшувати наступні значення: 0.3 мА при змінному струмі частотою 50 Гц, 0.4 при змінному струмі частотою 400Гц і 1 мА при постійному струмі. При високій температурі і великій вологості (відносна вологість більша 75%) допустимий струм слід зменшити в 3 рази.

Найбільш небезпечним для людини є струм частотою 20-200 Гц. З підвищенням або пониженням частоти небезпека зменшується і повністю зникає при частотах вище 400кГц, хоча високочастотні напруги можуть викликати опіки. Постійний струм є менш небезпечний лише при напругах менше 300 В, при дальшому збільшенні напруги небезпека постійного струму росте і при напрузі 600 В він є значно небезпечнішим за змінний струм.

На результати ураження впливає шлях проходження струму в тілі людини. Найбільш небезпечні випадки протікання струму через голову та грудну клітку людини.

- Порушення кровопостачання.

Виникає внаслідок незручного робочого положення в ногах,в руках та кистях.

- Запамороченння.

Що викликається перенапруженням зору, електромагнітним випромінюванням, неправельною організацією робочого місця.

- Головні болі, зашемлення нервових закінчень, подразнення шкіри та інші.

10.2.3.Аналіз методів дослідження, обладнання тахарактеристика речовин

Під час виконання бакалаврської роботи використовують персональні комп’ютери та периферійні пристрої (лазерні та струменеві друки, копіювальну техніку, сканери). Негативний вплив цих пристроїв на організм людини виникає через неадекватне (надто велике або надто мале) навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і прояву стресових реакцій. Найбільшому ризику виникнення різноманітних порушень піддаються: органи зору, м’язово-скелетна система, нервово-психічна діяльність, репродуктивна функція у жінок. Робота з комп’ютером характеризується значноюрозумовою напругою і нервово-емоційним навантаженням, високою напруженістю зорової праці та досить великим навантаженням на м’язи рук під час роботи з клавіатурою. У процесі роботи з комп’ютером необхідно дотримуватися правильного режиму праці та відпочинку. Інакше у людини відзначаються значна емоційна напруга зорового апарату, що може призвести до появи головного білю,дратівливості, порушення сну, почуття виснаження. Раціональний режим праці та відпочинку передбачає запровадження регламентованих перерв, рівномірний розподіл навантаження протягом робочого дня, регулярні комплекси вправ для очей, рук, хребта для поліпшення мозкового кругообігу та психофізіологічного розвантаження. З метою запобігання перевантаження організму як в цілому, так і окремих його функціональних систем, передусім зорового та рухового аналізаторів, центральної нервової системи, загальний час щоденної роботи з відеодисплейним терміналом треба обмежити чотирма годинами та обов’язково дотримуватись регламентованих перерв. Робота з ПК супроводжується також виділенням значної кількості тепла, шуму, вібрації та електромагнітних

випромінювань.

Уся практична частина роботи виконувалася на ПЕОМ класу IBMPC, з такими основними характеристиками:

1. Процесор – Intel Core i5 3450
2. Оперативна пам’ять – DDR3 8Gb
3. Жорсткий диск – WD 750Gb (WD7500AARS)
4. Монітор – LG L194WT
5. Операційна система – Windows 7 SP1

10.3.Організаційно-технічні заходи з поліпшення умов праці

10.3.1.Організування місця праці та безпечної роботи

В першу чергу, робоче місце користувача ПЕОМ повинно бути зручним для роботи. Організація робочого місця повина забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ГОСТ 12.2.032 “ССБТ. Рабочее место при выполнении работ сидя. Общие ергономические требования”; характеру та особливостям трудової діяльності.

Площа, виділена для одного робочого місця з відео терміналом, або персональною ПЕОМ, повина складати не менше 6 $м^2$, а обсяг - не менше 20 $м^3$. Робочі місця з відео терміналами відносно світлових прорізів повинні розташовуватись так, щоб природне світло падало збоку(переважно зліва).

Конструкція робочого місця користувача відео термінала (при роботі сидячи) має забезпечувати підтримання оптимальної робочої пози: ступні ніг – на підлозі(або на підставці для ніг); стегна – в горизонтальній площині; передпліччя – вертикально; лікті – під кутом 70-90 град. До вертикальної площини; зап’ястя – зігнуті під кутом не більше 20 град. відносно горизонтальної площини; нахил голови – 15-20 град.

Висота робочої поверхні столу для відео термінала має бути в межах 680-800 мм. Рекомендовані розміри столу: висота - 725 мм, ширина - 600-1400 мм, глибина – 800-1000 мм. Робочий стіл для відео термінала, як правило,має бути обладнаним підставкою для ніг шириною не менше 300 мм та глибиною не менше 400 мм, з можливістю регулювання по висоті в межах 150 мм та кута

нахилу опорної поверхні – в межах 20 град.

Робоче сидіння користувача відео термінала та персональної ЕОМ повинно мати такі основні елементи: сидіння, спинку та стаціонарні або зміні підлокітники. Робоче сидіння повинно бути підйомно-поворотним,таким, що регулюється за висотою, кутом нахилу сидіння та спинки, за відстанню спинки до переднього краю сидіння, висотою підлокітників. Ширина та глибина сидіння повинні бути не менше за 400 мм .Висота поверхні сидіння має регулюватися в межах 400-500 мм,а кут нахилу поверхні - від 15 град. вперед до 5 град. назад. Висота спинки повинна становити 300±20 мм, ширина – не менше 380мм, радіус кривизни по горизонтальній площині – 400 мм. Поверхня сидіння, спинки та підлокітників має бути напівм’ягкою, з неслизьким, не наелекризованим, повітронепроникним покриттям та забезпечувати можливість чищення від бруду.

Екран відео термінала та клавіатура мають розташовуватися на оптимальній відстані від очей користувача, але не ближче 600 мм, з урахуванням розміру алфавітно-цифрових знаків та символів. Розташування екрана відео термінала має забезпечувати зручність зорового спостереження у вертикальній площині під кутом ±30 град. від лінії зору працівника.

Клавіатуру слід розміщувати на поверхні столу або на спеціальній, регульованій за висотою,робочій поверхні окремо від столу на відстані 100-300 мм від краю, ближчого до працівника.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати дору видимість екрана відео термінала, зручність ручного керування пристроєм введення-виведеня інформації в зоні досяжності моторного поля: по висоті 900-1300 мм, по глибині 400-500 мм. Під матричні принтери слід підкладати вібраційні килимки для гасіння вібрації та шуму.

Виробниче середовище, яке є предметним оточенням людини, повинно суміщати в собі раціональне архітектурно-планувальне рішення, оптимальні санітарно-гігієнічні умови, легко-сприймаючий колір.

Відповідно до правил техніки безпеки при роботі з ЕОМ з’являються наступні вимоги.

1.До роботи допускаються особи, які пройшли інструктаж з техніки безпеки;

2.Перед початком роботи:

- необхідно привести до порядку робоче місце, звільнивши робочу зону від всього зайвого. На робочому місці все повинно бути розташоване в суворому порядку відповідно до послідовності виконуваних робіт;
- перевірити, чи достатньо освітлене робоче місце. При недостатній освітленості слід користуватись світильником місцевого освітлення, причому світло повинно падати зліва.

3. Під час роботи:

- робочий рукописний текст повинен бути розбірливим;
- сидіти потрібно прямо, не напружуючись і не налягаючи на край столу;
- для зручності слід опиратися на опорні планки для ніг;
- не вдаватись до зайвих розмов і відволікаючих шумів;
- тримати робоче місце необхідно в порядку і чистоті;
- відстань від паперу до очей повинна бути не менша 30 см, лікті повинні вільно лежати на столі, ні в що не впираючись;
- робота виконується з додатковими перервами через кожних 2 год. роботи перерва 5-10 хв.;
- регламентовані перерви слід заповнювати гімнастичними вправами з метою покращення кровопостачання мозку та кінцівок;
- при погіршенні зору або при інших симптомах погіршення стану здоров'я слід негайно звернутися до лікаря.

4. Не рекомендується:

- при роботі сидіти спиною до вікна;
- працювати при недостатній освітленості;

- працювати лише при місцевому освітленні;
- користуватися відкритим вогнем, палити, запалювати сірники у приміщенні;
- захаращувати робоче місце, підвіконня і проходи.

5.Після закінченні роботи слід прибрати і привести до ладу робоче місце.

10.3.2.Санітарно-гігієнічні вимоги до умов праці

Для підвищення працездатності та збереження здоров'я важливо утворити для організму людини стабільні кліматичні умови. До них належать: температура, відносна вологість, швидкість руху повітря та інтенсивність світлового випромінювання. З метою усунення з робочого приміщення забрудненого повітря і подачі чистого, здійснюється природна вентиляція за рахунок різниці температур всередині приміщення і ззовні. Відносна вологість повітря в приміщенні змінюється в межах 40-60 %. В лабораторії використовується природне бокове освітлення вдень, а ввечері або при недостатньому природному освітленні – штучне загальне освітлення, що рівномірно освітлює приміщення.

Слід пам'ятати, що при 8-ми годинному робочому дні допускається не більше ніж 4 години за ЕОМ. При цьому щогодини треба робити перерви на 5-10 хвилин, що дві години - на 15 хвилин.

10.3.3.Заходи щодо безпеки під час виконання дипломної(кваліфікаційної) роботи та професійної діяльності

Для забезпечення безпеки від ураження електричним струмом обслуговуючий персонал повинен систематично перевіряти: справність усіх частин електроустановки; стан ізоляції струмоведучих частин та захисного заземлення; справність блокування рубильників та інших пускових пристроїв. Безпека обслуговування електропристроїв забезпечується шляхом використання заземлення корпусів та елементів установок, які можуть виявитися під електричною напругою, надійного та швидкого захисного відключення при однофазному закороченні на землю частин електрообладнання або частини пошкодженої ділянки сітки, коли пристрій заземлення не забезпечує безпеку

обслуговування електропристрою.

У вимірювальних приладах, радіопристроях, апаратурі автоматики та обчислювальній техніці слід використовувати блокові схеми. Окремі блоки, встановлені в загальному корпусі, з'єднані між собою і з блоком живлення штепсельними розняттями. При висуванні блока штепсельний розняття закорочується і блок автоматично відключається від живлення, чим і забезпечується недоступність.

Крім того необхідно маркувати частини електрообладнання, наносити попереджувальні сигнали, написи та таблички.

Для захисту від електронного випромінювання необхідно на кожен дисплей ставити захисний екран, або замінювати електронно-променеві трубки рідкокристалічними матрицями.

Одним із найбільш ефективних та часто використовуваних методів захисту від радіоопромінення є екранування. Для екранів використовуються головним чином матеріали з великою електричною провідністю (мідь, латунь, алюміній та його сплави, сталь). Екрани повинні бути заземленні. Іншими мірами захисту є віддалення робочого місця від джерела електромагнітного випромінювання, раціональне розміщення в робочому приміщенні обладнання, випромінюючого електромагнітну енергію.

Основним методом боротьби з виробничим шумом є його зменшення в джерелі виникнення, звукопоглинання. Зменшити шум в джерелі виникнення можна за рахунок точного виготовлення окремих вузлів машини, зменшення зазорів.

Заходи безпеки під час експлуатації персонального комп’ютера та периферійних пристроїв передбачають:

- правильну організацію робочого місця та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;
- експлуатацію сертифікованого обладнання;
- дотримання заходів електробезпеки;
- забезпечення оптимальних параметрів мікроклімату;

- забезпечення раціонального освітлення робочого місця;
- зниження рівня шуму та вібрації.

10.4.Безпека в надзвичайних ситуаціях

10.4.1.Протипожежні та противибухові заходи

Основними причинами виникнення пожеж в лабораторіях є:

- необережне поводження з вогнем і з нагрівальними приладами;
- несправність електроустаткування, освітлення, виробничого устаткування і порушення правил їх експлуатації;
- самозапалювання горючих речовин;
- вибух парів і газів.

Всі організаційні і технічні протипожежні заходи мають на меті не допустити виникнення пожеж з будь-яких причин, запобігти поширенню пожеж в разі їх виникнення, забезпечити можливість евакуації людей і матеріальних цінностей із приміщень, що загорілись, якомога швидше ліквідувати пожежу.

Протипожежна профілактика полягає в недопущенні і ліквідації пожеж. На робочих місцях забороняється куріння та запалювання вогню. У випадку виникнення пожежі в електроустановках, що знаходяться під напругою, гасіння проводиться за спеціальною інструкцією. Для цієї мети використовують вуглекислі вогнегасники ВВ-2, ВВ-5, ВВ-8 місткістю 2, 5 і 8 літрів відповідно, а також вода, порошкові складники, накидки, сухі солі, пісок. В лабораторії знаходиться вогнегасник ВП-1. При користуванні вогнегасником слід пам’ятати про можливість утворення опіків від струменя піни.

У лабораторії також обов’язковим є присутність медичної аптечки з метою наданння першочергової медичної допомоги особі, яка отримала травму під час виробничого процесу.

10.4.2.Організування евакуації працівників

Виконуючи бакалаврську роботу було проаналізовано схему шляхів евакуації, які забезпечують якнайшвидше і найбезпечніше виведення людей з небезпечних зон. Проведення організованої евакуації з виробничих та інших приміщень і будівель, запобігання проявам паніки і недопущення загибелі

людей забезпечують шляхом складання плану евакуації з розробленням схеми евакуаційних шляхів та виходів. На підприємстві має бути встановлено порядок оповіщення людей про пожежу, з яким необхідно ознайомити всіх працівників. Серед загальних вимог до евакуаційних шляхів та виходів необхідно відмітити, що ними можуть бути дверні отвори, якщо вони ведуть з приміщень:

- безпосередньо назовні;
- на сходовий майданчик з виходом назовні безпосередньо або через вестибюль;
- у прохід або коридор з безпосереднім виходом назовні або на сходовий майданчик;
- у сусідні приміщення того ж поверху, що не містять виробництв, які належать за вибухопожежною та пожежною небезпекою до категорій А, Б і В та мають безпосередній вихід назовні або на сходовий майданчик.

Для безпечної евакуації шляхи та виходи мають відповідати таким вимогам:

- евакуаційні шляхи і виходи повинні утримуватися вільними, не захаращуватися та у разі потреби забезпечувати евакуацію всіх людей, які перебувають у приміщеннях;
- кількість та розміри евакуаційних виходів, їхні конструктивні рішення, умови освітленості, забезпечення незадимленості, протяжність шляхів евакуації, їхнє оздоблення повинні відповідати протипожежним вимогам будівельних норм;
- у приміщенні, яке має один евакуаційний вихід, дозволяється одночасно розміщувати не більше 50 осіб, а у разі перебування в ньому понад 50 осіб по-винно бути щонайменше два виходи, які відповідають вимогам будівельних норм.

Висновки

В дипломній роботі описано методи критичного шляху (CPM) і PERT(Project Evaluation and Review Technique) та програмні середовища, у яких виконана дипломна робота.

Згідно із проведеним аналізом стану умов праці встановлено, що санітарно-гігієнічні та ергономічні умови праці відповідають нормативам і їх не треба покращувати.

Написано програму для реалізації повнофункціонального методу PERT. Програма написана мовою C# у середовищі розробки Microsoft Visual Studio 2010.

Створено зручний користувацький інтерфейс, який:

включає форми для внесення вхідних даних проекту;

дозволяє обраховувати критичний шлях проекту та показує, які саме роботи належать до критичного шляху;

реалізує обчислення очікуваної тривалості та відхилення кожної роботи та проекту в цілому;

обраховує оцінку ймовірності виконання проекту за деякий заданий час;

реалізує запис результатів роботи у базу даних;

надає можливість внесення робіт за допомогою матриць суміжності мережевого графа;

включає зручні налаштування : відображення потрібних характеристик робіт, можливість зміни вагових коефіцієнтів оцінок тривалості робіт, можливість задання точної тривалості робіт.

Проведено апробацію результатів роботи програми за допомогою аналізу мережевого графіку деякого проекту, точна тривалість виконання робіт якого невідома. Проект включає у себе шість робіт, що характеризуються оптимістичною, найбільш ймовірною та песимістичною оцінками тривалості. Було реалізовано обчислення параметрів заданого проекту за допомогою методів PERT та CPM, і перевірено правильність отриманих результатів.

Список використаної літератури

1. Троелсен, Эндрю. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2013. — 1312 с.
2. Экономико-математические методы и прикладные модели: Учеб. пособие для вузов / В.В. Федосеев, А.Н. Гармаш, Д.М. Дайитбегов и др.; Под ред. В.В. Федосеева. - М.: ЮНИТИ, 1999. – 391 с.
3. Ноздріна Л. В., Ящук В. І., Полотай О. І. Управління проектами: Підручник / За заг. ред. Л. В. Ноздріної. — К.: Центр учбової літератури, 2010. — 432 с.
4. Дослідження та розробка моделей та алгоритмів нечіткого управління проектами на основі нечіткої логіки [Електронний ресурс]: – Режим доступу: http://masters.donntu.edu.ua/2012/iii/yakimova/diss/indexu.htm
5. System.Data.OleDb – простір імен [Електронний ресурс]: – Режим доступу: http://msdn.microsoft.com/ru-ru/library/system.data.oledb.aspx
6. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полный справочник = SQL: The Complete Reference. — 3-е изд. — М.: Вильямс, 2010. — 960 с.
7. Основи побудови мережевого графіка та його аналіз [Електронний ресурс]: – Режим доступу: http://wiki.fizmat.tnpu.edu.ua/index.php/Лекція:_Основи_побудови_мережевого_графіка_та_його_аналіз.
8. Управління спецпроектами (конспект лекцій НУДПСУ) [Електронний ресурс]: – Режим доступу: http://studentbooks.com.ua/content/view/1307/42/1/0/
9. Навакітян О. О., Кальниш В. В., Стрюков С. Н. Охорона праці користувачів відеодисплейних терміналів. – Київ: Основа, 2007. – 400 с.
10. Пожежна безпека. Нормативні акти та інші документи. В 2–х т.– Київ: Основа. – 2007. – Т.1 – 446 с., Т.2 – 448 с.

Додаток. Код програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;

namespace Method_PERT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            for (int i = 5; i <= 12; i++)
            {
                dataGridView1.Columns[i].DefaultCellStyle.BackColor = Color.Cyan;
            }
            dataGridView2.Columns[1].DefaultCellStyle.BackColor = Color.Cyan;
            dataGridView2.Columns[2].DefaultCellStyle.BackColor = Color.Cyan;
        }

        public string[] ComandCollection = new string[10];
        public string GetSQL = "";
        public string ConnectionToDataBase = "";

        public struct acts
        {
            public int start, finish;
            public double optimistic, pessimistic, mostlikely, duration, sigma;
            public double earlyStart, earlyFinish, lateStart, lateFinish;
            public double fullTimeReserve;
            public bool isCritical;
        };

        acts[] Act = new acts [100];

        int minV, maxV;
        double critWay, totalSigma;
        bool vidomoTrivialist;
        int optKoef=1, pessKoef=1, mostKoef=4, totalKoef=6;

        public void Zchit (acts[] Act)
        {
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                Act[i].start = Convert.ToInt32(dataGridView1[0, i].Value);
                Act[i].finish = Convert.ToInt32(dataGridView1[1, i].Value);
                Act[i].optimistic = Convert.ToInt32(dataGridView1[2, i].Value);
                Act[i].mostlikely = Convert.ToDouble(dataGridView1[3, i].Value);
                Act[i].pessimistic = Convert.ToDouble(dataGridView1[4, i].Value);
            }
            minV = Act[0].start;
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                if (Act[i].start < minV) minV = Act[i].start;
                if (Act[i].finish > maxV) maxV = Act[i].finish;
            }

        }
```

```
        public void SaveInBd()
        {
            string connectionString1 = "Provider=Microsoft.Ace.OLEDB.12.0;" + @"Data Source=
../../../BD1.accdb";
            OleDbConnection connection1 = new OleDbConnection(connectionString1);
            OleDbCommand command1 = connection1.CreateCommand();
            command1.CommandText = "DELETE FROM СписокРобіт ;";
            connection1.Open();
            command1.ExecuteNonQuery();
            connection1.Close();

            string connectionString = "Provider=Microsoft.Ace.OLEDB.12.0;" + @"Data Source=
../../../BD1.accdb";
            OleDbConnection connection = new OleDbConnection(connectionString);
            OleDbCommand command = connection.CreateCommand();
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                string val1 = (string) dataGridView1[0,i].Value;
                string val2 = (string) dataGridView1[1, i].Value;
                string val3 = (string)dataGridView1[2, i].Value;
                string val4 = (string)dataGridView1[3, i].Value;
                string val5 = (string)dataGridView1[4, i].Value;
                double val6 = (double)dataGridView1[5, i].Value;
                double val7 = (double)dataGridView1[6, i].Value;
                double val8 = (double)dataGridView1[7, i].Value;
                double val9 = (double)dataGridView1[8, i].Value;
                double val10 = (double)dataGridView1[9, i].Value;
                double val11 = (double)dataGridView1[10, i].Value;
                double val12 = (double)dataGridView1[11, i].Value;
                bool val13 = (bool)dataGridView1[12, i].Value;
                command.CommandText = "INSERT INTO СписокРобіт (IdПочаткуРоботи,
IdКінцяРоботи, ОптимістичнаОцінка, НайбільшймовірнаОцінка, ПесимістичнаОцінка, Тривалість,
СереднєВідхилення, РаннійСтарт, ПізнійСтарт, РаннійФініш, ПізнійФініш, РезервЧасу,
КритичнийШлях) VALUES('" + val1 + "','" + val2 + "','" + val3 + "','" + val4 + "','" + val5
+ "','" + val6 + "','" + val7 + "','" + val8 + "','" + val9 + "','" + val10 + "','" + val11
+ "','" + val12 + "','" + val13 + "');";
                connection.Open();
                command.ExecuteNonQuery();
                connection.Close();
            }



            string connectionString2 = "Provider=Microsoft.Ace.OLEDB.12.0;" + @"Data Source=
../../../BD1.accdb";
            OleDbConnection connection2 = new OleDbConnection(connectionString2);
            OleDbCommand command2 = connection2.CreateCommand();
            command2.CommandText = "DELETE FROM ЙмовірністьВиконання ;";
            connection2.Open();
            command2.ExecuteNonQuery();
            connection2.Close();

            string connectionString3 = "Provider=Microsoft.Ace.OLEDB.12.0;" + @"Data Source=
../../../BD1.accdb";
            OleDbConnection connection3 = new OleDbConnection(connectionString3);
            OleDbCommand command3 = connection3.CreateCommand();
            for (int i = 0; i < dataGridView2.RowCount - 1; i++)
            {
                string val1 = (string)dataGridView2[0, i].Value;
                double val2 = (double)dataGridView2[1, i].Value;
                double val3 = (double)dataGridView2[2, i].Value;
```

```
                command.CommandText = "INSERT INTO ЙмовірністьВиконання  (ПрогнозованийЧас,
Запізнення, ЙмовірністьВиконання) VALUES('" + val1 + "','" + val2 + "','" + val3 + "');";
                connection.Open();
                command.ExecuteNonQuery();
                connection.Close();
            }
        }

        public void Average (acts[] Act)
        {
            optKoef =  (int)numericUpDown2.Value;
            mostKoef = (int)numericUpDown3.Value;
            pessKoef = (int)numericUpDown4.Value;
            totalKoef = optKoef+mostKoef+pessKoef;
                for (int i = 0; i < dataGridView1.RowCount - 1; i++)
                {
                    if (!vidomoTrivalist)
                    {
                        Act[i].duration = (optKoef * Act[i].optimistic + mostKoef *
Act[i].mostlikely + pessKoef * Act[i].pessimistic) / totalKoef;
                        Act[i].sigma = (pessKoef * Act[i].pessimistic - optKoef *
Act[i].optimistic) / totalKoef;
                    }
                    if (vidomoTrivalist)
                    {
                        Act[i].duration = Convert.ToDouble(dataGridView1[5, i].Value);
                        Act[i].sigma = 0;
                    }

                    dataGridView1[5, i].Value = Act[i].duration;
                    dataGridView1[6, i].Value = Act[i].sigma;
                }

        }

        public void Critical_Path_Method(acts[] Act)
        {


            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                Act[i].earlyStart = 0;
                Act[i].earlyFinish= 0;
                for (int j = 0; j < dataGridView1.RowCount - 1; j++)
                {
                    if (Act[j].finish == Act[i].start &&
Act[j].earlyFinish>Act[i].earlyStart)
                    {
                        Act[i].earlyStart = Act[j].earlyFinish;
                    }
                }
                Act[i].earlyFinish = Act[i].earlyStart + Act[i].duration;
                dataGridView1[7, i].Value = Act[i].earlyStart;
                dataGridView1[8, i].Value = Act[i].earlyFinish;
            }



            for (int i = dataGridView1.RowCount - 2; i >= 0; i--)
            {
                Act[i].lateStart = 0;
                Act[i].lateFinish = 0;
                for (int j = dataGridView1.RowCount - 2; j >= 0 ; j--)
                {
```

```
                    if (Act[j].finish == Act[i].finish && Act[i].finish == maxV &&
Act[j].earlyFinish > Act[i].lateFinish)
                    {
                        Act[i].lateFinish = Act[j].earlyFinish;

                    }

                    else
                     if (Act[i].lateFinish ==0)   Act[i].lateFinish = Int32.MaxValue;
                    if (Act[j].start == Act[i].finish && Act[j].lateStart <
Act[i].lateFinish)
                    {
                        Act[i].lateFinish = Act[j].lateStart;

                    }

                }
                Act[i].lateStart = Act[i].lateFinish - Act[i].duration;
                if (Act[i].lateStart < 0) Act[i].lateStart = 0;
                dataGridView1[9, i].Value = Act[i].lateStart;
                dataGridView1[10, i].Value = Act[i].lateFinish;
            }


            critWay = 0;
            totalSigma = 0;
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                Act[i].fullTimeReserve = Act[i].lateFinish - Act[i].earlyFinish;
                if (Act[i].fullTimeReserve < 0) Act[i].fullTimeReserve = 0;
                if (Act[i].fullTimeReserve == 0)
                {
                    Act[i].isCritical = true;
                    critWay += Act[i].duration;
                    totalSigma += Math.Pow(Act[i].sigma,2);
                    dataGridView1[12, i].Style.ForeColor = Color.Red;
                    dataGridView1[12, i].Style.Font = new Font(dataGridView1.Font,
FontStyle.Bold);
                }
                dataGridView1[11, i].Value = Act[i].fullTimeReserve;
                dataGridView1[12, i].Value = Act[i].isCritical;
            }
            totalSigma = Math.Sqrt(totalSigma);
            textBox1.Text = critWay.ToString();
            textBox2.Text = totalSigma.ToString();

        }



        public void SearchProbability()
        {
            double z, p,ts;
            p = 0;
            for (int i = 0; i < dataGridView2.RowCount - 1; i++)
            {
                ts = Convert.ToDouble(dataGridView2[0, i].Value);
                z = (ts - critWay) / totalSigma;
                if (z < -1.5) p = 0.02;
                if (z >= -1.5 && z<-1) p = 0.07;
                if (z >= -1 && z < -0.7) p = 0.16;
                if (z >= -0.7 && z < -0.5) p = 0.24;
                if (z >= -0.5 && z < -0.3) p = 0.31;
                if (z >= -0.3 && z < -0.1) p = 0.36;
                if (z >= -0.1 && z < 0.1) p = 0.38;
```

```
            if (z >= 0.1 && z < 0.3) p = 0.54;
            if (z >= 0.3 && z < 0.5) p = 0.62;
            if (z >= 0.5 && z < 0.7) p = 0.69;
            if (z >= 0.7 && z < 1.0) p = 0.76;
            if (z >= 1.0 && z < 1.5) p = 0.84;
            if (z >= 1.5 && z < 2) p = 0.93;
            if (z >= 2) p = 0.98;
            dataGridView2[1, i].Value = z;
            dataGridView2[2, i].Value = p;
        }
    }

    public void AddAct()
    {
        int i = dataGridView1.RowCount-1;
        dataGridView1.RowCount++;
        dataGridView1[0, i].Value = textBox3.Text;
        dataGridView1[1, i].Value = textBox4.Text;
        if (!vidomoTrivalist)
        {
            dataGridView1[2, i].Value = textBox5.Text;
            dataGridView1[3, i].Value = textBox6.Text;
            dataGridView1[4, i].Value = textBox7.Text;
        }
        if (vidomoTrivalist)
        {
            dataGridView1[5, i].Value = textBox8.Text;
        }
    }

    public void PidpisKomirok(DataGridView dataGridView3)
    {
        int N= (int) numericUpDown1.Value;
        dataGridView3.ColumnCount = dataGridView3.RowCount = N;
        dataGridView3.RowHeadersWidth = 50;
        for (int i = 0; i < N; i++)
        {
            dataGridView3.Columns[i].HeaderText = (i + 1).ToString();
            dataGridView3.Columns[i].Width = 30;
            dataGridView3.Rows[i].HeaderCell.Value = (i + 1).ToString();
        }
    }

    public void ZchitMatr1()
    {
        int k=0;
        dataGridView1.RowCount = 1;
        for (int i = 0; i < dataGridView3.RowCount - 1; i++)
        {
            for (int j = 0; j < dataGridView3.RowCount - 1; j++)
            {
                if (Convert.ToInt32(dataGridView3[j, i].Value) > 0 ||
Convert.ToInt32(dataGridView4[j, i].Value) > 0 || Convert.ToInt32(dataGridView5[j, i].Value)
> 0)
                {
                    dataGridView1.RowCount++;
                    dataGridView1[0, k].Value = i + 1;
                    dataGridView1[1, k].Value = j + 1;
                    dataGridView1[2, k].Value = dataGridView3[j, i].Value;
                    dataGridView1[3, k].Value = dataGridView4[j, i].Value;
                    dataGridView1[4, k].Value = dataGridView5[j, i].Value;
                    k++;
                }
            }
        }
```

```
        }

        public void ZchitMatr2()
        {
            int k = 0;
            dataGridView1.RowCount=1;
            for (int i = 0; i < dataGridView6.RowCount - 1; i++)
            {
                for (int j = 0; j < dataGridView6.RowCount - 1; j++)
                {
                    if (Convert.ToInt32(dataGridView6[j, i].Value) > 0)
                    {
                        dataGridView1.RowCount++;
                        dataGridView1[0, k].Value = i + 1;
                        dataGridView1[1, k].Value = j + 1;
                        dataGridView1[5, k].Value = dataGridView6[j, i].Value;
                        k++;
                    }
                }
            }
        }

        public void ClearMatr1()
        {
            int curRozmirMatr = (int)numericUpDown1.Value;
            numericUpDown1.Value = 1;
            numericUpDown1.Value = curRozmirMatr;
        }

        public void Clear2()
        {
            for (int i = 0; i < dataGridView2.RowCount - 1; i++)
            {
                for (int j = 0; j < dataGridView2.ColumnCount; j++)
                {
                    dataGridView2[j, i].Value = 0;
                }
            }
            dataGridView2.RowCount = 1;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                dataGridView1[12, i].Style.ForeColor = Color.Empty;
                dataGridView1[12, i].Style.Font = new Font(dataGridView1.Font,
FontStyle.Regular);
            }
            try
            {
                acts[] Act = new acts[dataGridView1.RowCount];
                Zchit(Act);
                Average(Act);
                Critical_Path_Method(Act);
            }
            catch { MessageBox.Show("Помилка! Перевірте правильність введених даних"); }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                Act[i].start = new int();
                Act[i].finish = new int();
```

```
                Act[i].optimistic =new double();
                Act[i].mostlikely = new double();
                Act[i].pessimistic = new double();
                Act[i].duration = new double();
                Act[i].sigma = new double();
                Act[i].earlyStart = new double();
                Act[i].earlyFinish = new double();
                Act[i].lateStart = new double();
                Act[i].lateFinish = new double();
                Act[i].fullTimeReserve = new double();
                Act[i].isCritical = new bool();
            }

            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                for (int j = 0; j < dataGridView1.ColumnCount; j++)
                {
                    dataGridView1[j, i].Value = 0;
                }
            }

            dataGridView1.RowCount = 1;
            textBox1.Text = "0";
            textBox2.Text = "0";
            minV = new int();
            maxV = new int();
            critWay = new double();


            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                    dataGridView1[12, i].Style.ForeColor = Color.Empty;
                    dataGridView1[12, i].Style.Font = new Font(dataGridView1.Font,
FontStyle.Regular);
            }

        }

        public void CheckedChanged(CheckBox checkBox1,int i)
        {
            if (checkBox1.Checked == true) dataGridView1.Columns[i].Visible = true;
            if (checkBox1.Checked == false) dataGridView1.Columns[i].Visible = false;
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            CheckedChanged(checkBox1, 5);
        }

        private void checkBox2_CheckedChanged(object sender, EventArgs e)
        {
            CheckedChanged(checkBox2, 6);
        }

        private void checkBox3_CheckedChanged(object sender, EventArgs e)
        {
            CheckedChanged(checkBox3, 7);
        }

        private void checkBox4_CheckedChanged(object sender, EventArgs e)
        {
            CheckedChanged(checkBox4, 8);
        }

        private void checkBox5_CheckedChanged(object sender, EventArgs e)
```

```
{
    CheckedChanged(checkBox5, 9);
}

private void checkBox6_CheckedChanged(object sender, EventArgs e)
{
    CheckedChanged(checkBox6, 10);
}

private void checkBox7_CheckedChanged(object sender, EventArgs e)
{
    CheckedChanged(checkBox7, 11);
}

private void checkBox8_CheckedChanged(object sender, EventArgs e)
{
    CheckedChanged(checkBox8, 12);
}

private void checkBox9_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox9.Checked == true)
    {
        vidomoTrivalist = true;
        dataGridView1.Columns[2].Visible = false;
        dataGridView1.Columns[3].Visible = false;
        dataGridView1.Columns[4].Visible = false;
        dataGridView1.Columns[5].ReadOnly = false;
        dataGridView1.Columns[5].HeaderText = "Тривалість операції";
        dataGridView1.Columns[5].DefaultCellStyle.BackColor = Color.Empty;
        checkBox2.Checked = false;

        panel7.Visible = false;
        panel8.Visible = false;
        panel9.Visible = false;
        panel10.Visible = true;
        panel10.Location = new Point(31, 41);

        panel4.Visible = false;
        panel5.Visible = false;
        panel6.Visible = false;
        panel11.Visible = true;
    }
    if (checkBox9.Checked == false)
    {
        vidomoTrivalist = false;
        dataGridView1.Columns[2].Visible = true;
        dataGridView1.Columns[3].Visible = true;
        dataGridView1.Columns[4].Visible = true;
        dataGridView1.Columns[5].ReadOnly = true;
        dataGridView1.Columns[5].HeaderText = "Середньозважений час операції";
        dataGridView1.Columns[5].DefaultCellStyle.BackColor = Color.Cyan;

        panel7.Visible = true;
        panel8.Visible = true;
        panel9.Visible = true;
        panel10.Visible = false;

        panel4.Visible = true;
        panel5.Visible = true;
        panel6.Visible = true;
        panel11.Visible = false;
    }
}
```

```
        private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                SearchProbability();
            }
            catch
            {
                if (vidomoTrivalist && totalSigma == 0) MessageBox.Show("Точна тривалість
проекту є відомою");
                else
                    MessageBox.Show("Помилка! Перевірте правильність введених даних");
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            try
            {
                AddAct();
                MessageBox.Show("Роботу додано до списку");
            }
            catch { MessageBox.Show("Помилка! Перевірте правильність введених даних"); }
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {
            PidpisKomirok(dataGridView3);
            PidpisKomirok(dataGridView4);
            PidpisKomirok(dataGridView5);
            PidpisKomirok(dataGridView6);
        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (!vidomoTrivalist) ZchitMatr1();
            if (vidomoTrivalist) ZchitMatr2();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            ClearMatr1();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            SaveInBd();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            Clear2();
        }


    }
}
```