МЕХАНІЧНИЙ ФАКУЛЬТЕТ

Кафедра вагонів

МЕТОДИЧНІ ВКАЗІВКИ

та завдання до курсової роботи
з дисципліни

«СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ РУХОМОГО СКЛАДУ»

Частина 2

Харків – 2015

Методичні вказівки розглянуто і рекомендовано до друку на засіданні кафедри вагонів» 10 лютого 2014 р., протокол № 7.

Рекомендовано для студентів денної і заочної форм навчання напряму підготовки 6.070105 «Рухомий склад залізниць».

Укладачі:

доц. В.С. Меркулов,
асистенти І.М. Афанасенко,
Д.І. Скуріхін

Рецензент

проф. І.Е. Мартинов

МЕТОДИЧНІ ВКАЗІВКИ ТА ЗАВДАННЯ

до курсової роботи
з дисципліни

«СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ РУХОМОГО СКЛАДУ»
Частина 2

Відповідальний за випуск Меркулов В.С.

Редактор Буранова Н.В.

Підписано до друку 06.05.14 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 2,0. Тираж 50. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 2874 від 12.06.2007 р.

Зміст

Вступ

Visual Basic — остання версія однієї з популярних мов програмування. У наш час за допомогою Visual Basic можна швидко створювати додатки, що працюють у середовищі Windows, для будь-якої галузі комп'ютерних технологій: бізнес-додатки, мультимедіа, додатки типу клієнт-сервер, додатки управління базами даних. Крім того, Visual Basic є вбудованою мовою для додатків Microsoft Office. Багато розробників також використовують Visual Basic як внутрішню мову своїх додатків.

Методичні вказівки містять стислий опис основних прийомів використання Visual Basic для виконання розрахункової частини курсової роботи з дисципліни «Системи автоматизованого проектування рухомого складу» та приклад розроблення відповідного проекту.

1 Мова програмування Visual Basic

1.1 Загальні відомості

Visual Basic (VB) — це універсальна мова програмування.

Сукупність програмних засобів, за допомогою яких створюються нові програми, називається середовищем програмування VB.

Середовище візуального програмування VB — це графічна автоматизована оболонка над об'єктно-орієнтованою версією мови **Basic**.

Це середовище містить набір інструментів, що полегшують і прискорюють процес розробки. Причому процес розробки полягає не в написанні програми (програмного коду). Оскільки це середовище поставляється разом з операційним середовищем Windows, то й створювати можливо програми, що називаються **Windows-Додатки** або просто **Додатки**.

Принципове нововведення VB полягало в реалізації ідей **подійно-керованого** та **візуального програмування** в середовищі Windows, які радикально відрізнялися від класичних схем розробки програм.

Додаток формується засобами графічного редагування (компонування), що дозволяє значно полегшити процес створення програмного коду.

Для створення додатка необхідно скласти проект.

Проектом називають сукупність файлів, що входять у додаток, зберігають інформацію про його компоненти і з яких VB створює готову для виконання програму.

Процес створення проекту (і, при бажанні, Windows - додатка) складається з таких етапів:

– створення екранної форми (з об'єктами і їхніми властивостями);

– написання програми (програмного коду);

– налагодження програми, тобто усунення в ній логічних помилок;

– перетворення проекту у Windows-додаток.

VB — **об'єктно-орієнтована мова**. Основою мови є об'єкти.

Об'єкт VB — комбінація програмного коду і даних, яку можна вважати одним цілим.

Об'єктом є частина додатка – форма або елемент управління у вікні: кнопки, списки, текстові поля тощо. Цілий додаток теж є об'єктом.

На відміну від традиційних поглядів, коли програму розглядали як набір підпрограм або як перелік інструкцій до комп'ютера, об'єктно-орієнтовану програму можна вважати сукупністю об'єктів, кожний з яких здатний отримувати повідомлення, обробляти дані та надсилати повідомлення іншим об'єктам.

Кожний об'єкт має:

– властивості;
– методи;
– події.

Властивості — це показники, що характеризують об'єкт.

Методи — це дії, які можна зробити з об'єктом.

Події — це дії, які відбуваються з об'єктом.

Властивістю об'єкта у VB є якісна або кількісна його характеристика: розміри, положення на екранній формі, кольори самого об'єкта, кольори тексту, поміщеного на об'єкт, характеристики шрифту і т. д.

Методом називається команда, що дається об'єкту.

Методи у VB — це програмні процедури (або фрагменти коду), які виконують деяку обробку, пов'язану з об'єктом. За допомогою методів можна наказати об'єкту виконати ті або інші дії.

Кожному діалоговому елементу у VB поставлений у відповідність певний набір подій, що відбуваються в період виконання програми.

Події виникають у результаті одержання повідомлень.

Щоразу, коли натискається кнопка, переміщається миша, змінюються розміри форми й т.д., Windows генерує повідомлення, що описує дії, і поміщає його в чергу повідомлень програми. Із черги повідомлення доставляється відповідному об'єкту, а той генерує відповідну подію.

У свою чергу, кожній події ставиться у відповідність процедура її обробки — набір операторів, виконуваних при виклику процедури. Можна написати свій власний фрагмент програми, у якому об’єкт буде реагувати на подію саме так, як потрібно.

1.2 Запуск середовища програмування Visual Basic

Перед роботою з VB його необхідно налаштувати. Для цього запустіть VB

Пуск=>Програми=>Microsoft Visual Basic 6.0=>
Microsoft Visual Basic 6.0.

Примітка – Установка VB не відрізняється особливою складністю, все стандартно. У процесі інсталяції необхідно вказати компоненти, які будуть установлені на комп’ютер.

*Необхідно зайти в меню **Tools=>Options**, встановити галочку "**Require Variable Declaration**". Це позбавить від зайвих помилок при автоматичному визначенні змінних.*

*Далі на вкладці **Editor Format**, у списку **Font** необхідно вказати **Courier New Cyr**. Якщо цього не зробити, то VB не буде коректно відображати кирилицю.*

*Також бажано встановити колір зарезервованих слів у яскраво-синій. Для цього вибрати в списку **Code Colors Keyword Text** і в полі **Foreground** указати яскраво-синій колір (сьомий знизу).*

При створенні форм розроблювач може використовувати шаблони та майстри, а також конструктор форм. Конструктор форм призначений для самостійної розробки форми із заданими властивостями або для зміни форми, створеної за допомогою майстра.

Створення будь-якого додатка у VB починається зі створення проекту. Щоб створити свій новий проект, необхідно запустити програму VB. З’явиться вікно **New Project** – новий

проект із трьома вкладками: **New** — нові проекти, **Existing** — існуючі проекти і **Recent** — проекти, що використовувалися недавно. За його допомогою можна створити новий проект або відкрити існуючий.

За замовчуванням обрана вкладка **New** і виділений значок **Standard EXE** (рисунок 1). Натисніть кнопку **Відкрити**.

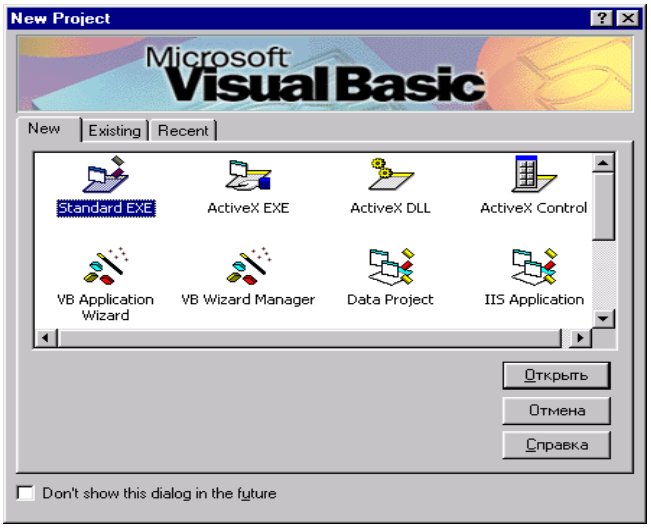
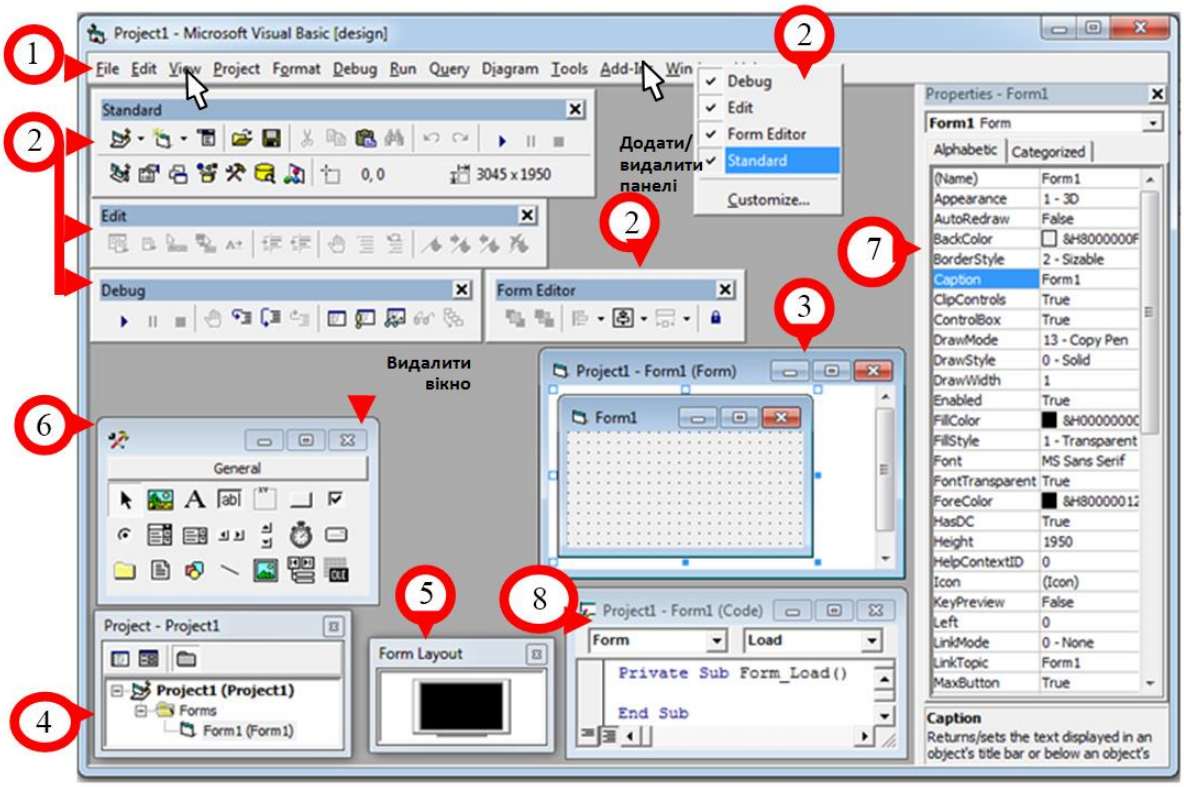


Рисунок 1 — Вікно відкриття проекту Visual Basic

1.3 Вікно середовища програмування Visual Basic

Після виконання попередніх дій відкривається вікно **Project** (рисунок 2), що містить нову форму, з якої можна починати працювати: змінювати встановлені за замовчуванням властивості, поміщати в неї елементи управління, використовуючи для цього панель елементів управління, що є основним робочим інструментом при розробленні форм додатка. У рядку заголовка цього вікна з'являється слово **design**, яке вказує, що програма перебуває в режимі розроблення додатка.



1 – головне Меню (Menu) — повний набір команд VB;

2 – панель інструментів (Toolbars) — команди головного меню, які найбільш часто використовуються.

Додати/видалити на екран панель інструментів можна **View=>Toolbars** або використовуючи **контекстне меню** в області панелі інструментів;

3 – вікно конструктора (редактора) форм (Form Designer) — розташовано в центрі та має заголовок **Project1 - Form1(Form).**

Додати на екран **View=>Object** або з клавіатури **Shift+F7;**

4 – вікно провідника проектів (Project Explorer) — призначено для роботи зі структурою проекту.

Додати на екран **View=>Project Explorer**, з клавіатури **Ctrl+R** або на панелі інструментів кнопка ;

5 – вікно макета розміщення форм (Form Layout) — використовується для зручності надання форми на екрані.

Додати на екран **View=>Form Layout Window** або на панелі інструментів кнопка ;

6 – палітра елементів управління (компонентів) (ToolBox або **General)** — містить об'єкти, які розташовуються на формі.

Додати на екран **View=>Toolbox** або на панелі інструментів кнопка ;

7 – вікно властивостей (Properties Window) — відображає властивості виділених об'єктів.

Додати на екран **View=>Properties Window**, з клавіатури **F4** або на панелі інструментів кнопка ;

8 – вікно коду (Code) — програмний код.

Додати на екран **View►Code**, контекстне меню об'єкта, для якого пишеться програмний код.

Рисунок 2 — Вікно середовища програмування Visual Basic

Вікно форми можна вивести на екран, двічі натиснувши у вікні провідника проекту по значку або імені форми.

1.4 Збереження проекту

Проект додатка зберігається в окремому файлі, в окремих файлах зберігаються елементи проекту.

При першому збереженні вказуються імена файлів для всіх елементів проекту. Оскільки проект складається з декількох файлів, то для нього краще створити окрему папку.

Порядок збереження проекту, що містить одну форму:

1 У меню **File** виберіть команду **Save Project** або натисніть кнопку **Save Project** на стандартній панелі інструментів.

2 У діалоговому вікні, що відкрилося, **Save File As** (рисунок 3) список **Тип файлу** містить значення **Form Files**, указуючи, що зберігається форма, що входить у додаток. Використовуючи список, що розкривається, **Папка**, виберіть папку, у якій буде збережена форма, потім у поле **Ім'я файлу** введіть ім'я форми й натисніть кнопку **Зберегти**.

3 Оскільки жодних компонентів, крім форми, новий додаток не містить, з'являється діалогове вікно **Save Project As** для збереження самого проекту. За замовчуванням у списку **Папка** обрана папка, у якій зберегли форму. Введіть найменування проекту й натисніть кнопку **Зберегти**.

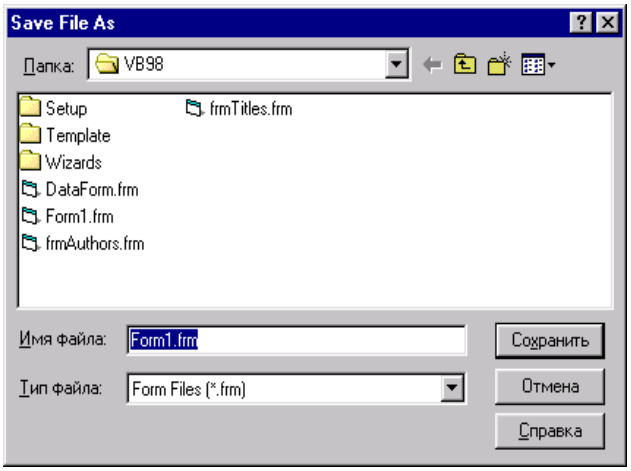


Рисунок 3 — Діалогове вікно Save File As для збереження файлів додатка

Примітка – При збереженні форми створюється файл із розширенням frm, у якому зберігається інформація про форму, її властивості, про розміщені в ній об'єкти і їхні властивості, а також заданий програмний код. Крім файлу з розширенням frm для форм, що містять графічні й інші бінарні об'єкти, створюється також файл із розширенням frx.

1.5 Відкриття проекту

Для того щоб одержати доступ до компонентів, що входять у проект, виконайте одну з таких дій:

1 У меню **View** виберіть команду **Project Explorer**.

2 Натисніть **кнопку Project Explorer** на стандартній панелі інструментів.

3 Натисніть комбінацію клавіш **<Ctrl>+<R>.**

4 Відкриється вікно провідника проекту (рисунок 4), що містить список усіх його компонентів.

1.6 Запуск і зупинка створюваного додатка

Для виконання створеного додатка існує багато способів:

1 Виберіть у меню **Run** команду **Start**.

2 Натисніть кнопку **Start**  на стандартній панелі інструментів.

3 Натисніть клавішу **<F5>.**

При цьому з'являється стартова форма, тобто вікно створюваної програми і його значок у панелі завдань.

1 Призупинити програму  Пауза (Ctrl + Pause Break) Використовується для налагодження.

2 Продовжити виконання програми:  або F5.

3 Зупинити програму Стоп  або закрити вікно запущеної програми  .

1.7 Робота з елементами середовища програмування

Елементи середовища програмування — це невеликі вікна, які виводять різну інформацію й дають змогу управляти складовими частинами проекту.

Вікна можна вивести на екран або сховати за допомогою меню **Вид**. Усі вікна середовища програмування можна перетягнути за заголовок до будь-якого краю екрана.

Провідник проекту — відображає групи об’єктів (наприклад: Форми, Модулі). У групах перебувають безпосередньо самі об’єкти: форми, модулі.

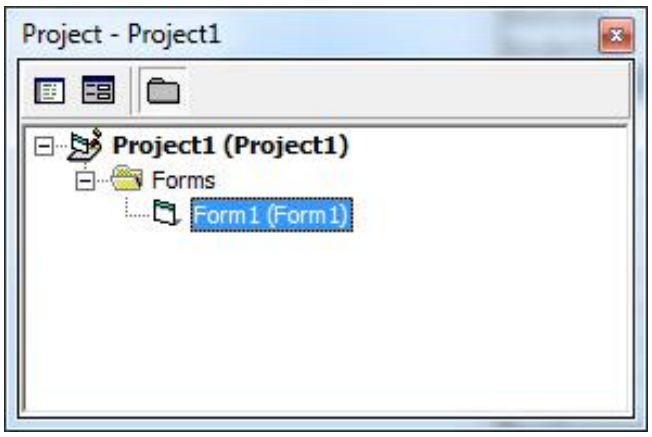


Рисунок 4 — Провідник проекту

У вікні після значка об’єкта вказується ім’я файлу, у якому він збережений.

– **Додати новий об’єкт** можна за допомогою меню **Project**.

Наприклад: додати форму: **Project => Add Form** (Додати форму) => Обрати шаблон форми => **Открыть**.

– **Вибрати об’єкт і відкрити його вікно:** у вікні провідника проекту двічі натиснути на назву потрібного об’єкта або натиснути кнопку — **View Object** (Показати об’єкт) у вікні провідника проекту.

– **Відкрити вікно коду об’єкта:** натиснути на назву об’єкту правою кнопкою й у контекстному меню вибрати рядок — **View Code** (Показати код) або натиснути кнопку у вікні Провідника проекту.

– **Видалити об’єкт із проекту:** клацнути правою кнопкою назву об’єкта й у контекстному меню вибрати **Remove Form (**Видалити**).**

Вікно властивостей (рисунок 5) — відображає властивості поточного об’єкта (форми або елементів управління: кнопок, списків, перемикачів).

Щоб зробити елемент поточним, необхідно натиснути по ньому.

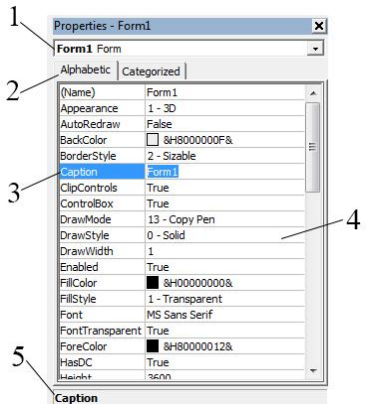


Рисунок 5 — Вікно властивостей

1 **Ім’я** об’єкта, властивості якого відображаються.

2 Вкладки: **Алфавіт і Категорії** — змінюють порядок сортування властивостей: за алфавітом, за категоріями (групами схожих властивостей).

3 Графа: **Назва властивості**.

4 Графа: **Значення властивості**.

5 **Коментар** поточної (обраної) властивості.

Значення властивості вписується із клавіатури або вибирається зі списку. Список значень відкривається кнопкою, кнопка із трьома крапками відкриває вікно діалогу, наприклад, для вибору файлів або шрифтів.

Значення властивостей можуть бути логічними, тобто мати значення:

True — тобто Так, Істина, 1.

False — тобто Ні, Хибність, 0.

Ці два значення змінюються подвійним натисканням по рядку потрібної властивості.

Вікно розміщення форми на екрані (рисунок 6) — показує, як буде розташована форма на екрані після запуску програми.

У цьому вікні на зображеному екрані монітора можна перетягувати форму мишею.

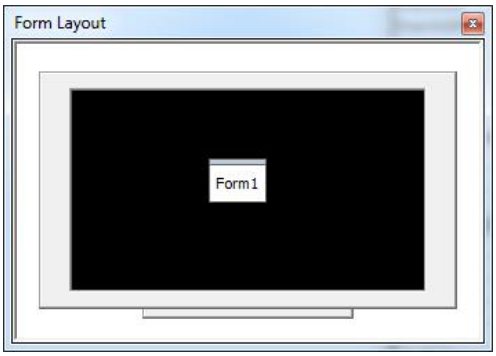


Рисунок 6 — Вікно розміщення форми

2 Форма

2.1 Загальні відомості

Форма — це ескіз вікна майбутньої програми (додатка).

Форма є об’єктом, тому має свої властивості, методи, події.

Форма є контейнером для інших об’єктів, тобто вона може містити кнопки, списки, текстові поля й тощо.

У проекті може міститися кілька форм. При запуску програми з’являється стартова форма. Для вибору стартової форми виконайте:

Project → Project Properties→ General (Головне) → **Startup Object** (Об’єкт запуску)**.**

Форми бувають:

1 **Форма** (Form) — звичайна форма, використовувана в нескладних програмах.

2 **Основна форма** (MDI Form) — це форма, що може містити дочірні (вкладені) форми. У додатку може бути тільки одна така форма.

3 **Дочірня форма** (Child) — міститься тільки усередині основної форми. Таких форм у додатку може бути кілька.

4 **Форма діалогу** (Dialog) — з'являється на екрані на короткий час, служить для уведення або виведення інформації, не змінюється в розмірах і перебуває поверх інших вікон.

2.2 Методи, події та властивості, характерні для форми

Атрибутами об'єкта Форма є близько 50 властивостей зазначеного об'єкта, які розташовуються у вікні властивостей.

Однією з найважливіших властивостей цього об'єкта є властивість **Name** (ім'я даного об'єкта). Форма може мати ім'я, відмінне від імені, заданого VB за замовчуванням.

Примітка – Змістовні імена полегшують роботу з формами та іменами програми.

Розглянемо деякі методи, події й властивості, характерні для Форми.

– **Load** — оператор, що завантажує форму в пам'ять, але не відображає її на екрані;

– **Unload** — оператор, що вивантажує форму з пам'яті й видаляє її з екрана;

– **Show** — метод, що завантажує й показує форму на екрані;

– **Hide** — метод, що видаляє форму з екрана, але не з пам'яті;

– **Activate** — подія, що відбувається, якщо форма стає активною;

– **Deactivate** — подія, що відбувається, якщо форма перестає бути активною;

– **Resize** — подія, що відбувається при зміні розмірів форми.

2.3 Зображення елементів управління на формі

Для організації інтерфейсу між користувачем і програмою служать елементи управління (рисунок 7).

Саме за допомогою кнопки користувач може дати команду програмі, що виведе результат своєї роботи у вигляді напису.

Створення форми з елементами управління є відповідальним етапом, тому що тут визначається не тільки, наскільки зручною буде програма для користувача, але й створюються об’єкти, які згодом будуть «оживати» у процесі програмування, будучи «каркасом» додатка.

Елементи управління додаються за допомогою панелі інструментів (Toolbars) (**Вид → Панель Інструментів**):

Рисунок 7 — Елементи управління на формі

– **Створити** (рисунок 8) елемент управління: виберіть потрібний елемент на панелі інструментів (натиснувши по кнопці елемента) і, втримуючи ліву кнопку, перемістіть покажчик миші по діагоналі на формі:

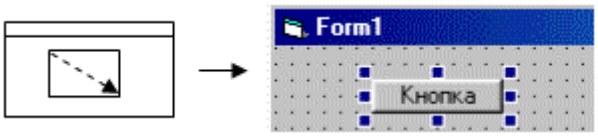


Рисунок 8 — Приклад створення елемента управління

– **Виділити** елемент управління: натисніть по потрібному елементу мишею. Після виділення елемента управління або форми у вікні властивостей можна змінити його властивості.

– **Видалити** елемент управління: виділіть елемент управління та натисніть клавішу Delete.

3 Елементи управління

Елементи управління — це об’єкти, які служать для організації інтерфейсу між користувачем і комп’ютером і мають свої властивості, методи, події (таблиця 1). Наприклад: кнопки, списки, перемикачі. На рисунку 9 наведений приклад використання елементів управління при розробленні програми.

Таблиця 1 — Елементи управління

Кнопка	Назва	Призначення
1	2	3
	**ComboBox (**Поле зі списком)	Створює на формі об'єкт, що містить одночасно поле введення й список, що розкривається
	**ListBox (**Список**)**	Створює на формі список для вибору одного або декількох значень із пропонованого списку значень
	**HScrollBar (**Горизонтальна смуга прокручування**)**	Розміщає у формі горизонтальну смугу прокручування, яка використовується як покажчик для вибору значення із заданого діапазону

Продовження таблиці 1

1	2	3
	**VScrollBar (**Вертикальна смуга прокручування)	Розміщує на формі вертикальну смугу прокручування, яка використовується як покажчик для вибору значення із заданого діапазону
	**Timer (**Таймер**)**	Розміщує на формі таймер
	**DriveListBox (**Список пристроїв**)**	Створює на формі список пристроїв
	**DirListBox** (Список папок)	Створює на формі деревоподібний список папок
	**FileListBox (**Список файлів)	Створює на формі список файлів
	**Shape** (Обрис)	Створює у формі геометричні фігури, такі як прямокутник, квадрат, коло, еліпс, прямокутник і квадрат з округленими кутами
	**Line** (Лінія)	Створює лінії
	**Image** (Зображення)	Створює у формі поля, призначені для відображення графічних зображень
	**Data** (Дані)	Створює елемент управління даними в базі даних для переміщення по записах і відображення результату навігації

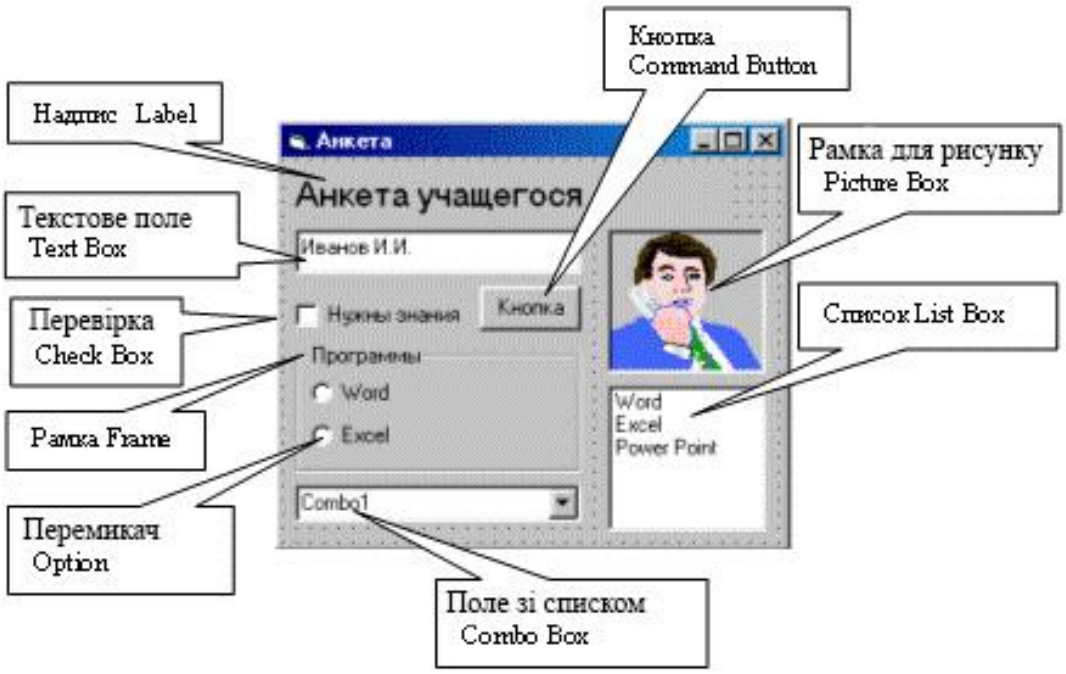


Рисунок 9 — Приклад використання елементів управління

Надпис **Label** — служить для додавання тексту на форму. Цей текст не може бути змінений користувачем програми, але може бути змінений розробником програми під час проектування або програмно.

Властивості:

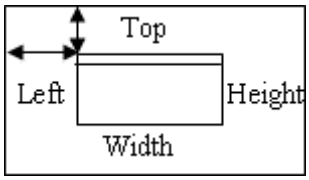
– **Caption** — текст напису;

– **Font —** шрифт, його розмір, накреслення;

– **Alignment** — вирівнювання тексту: **Left** (ліворуч), **Right** (враворуч), **Center** (по центру);

– **ToolTipText** — підказка, що з’являється при наведенні покажчика миші на елемент управління;

– **Властивості розміщення й розміру елемента:**



– **ForeColor** — колір тексту;

– **BackColor** — колір тіла.

Примітка – Можна вибирати системні кольори, що відповідають оформленню Windows, або будь-які з палітри.

Текстове поле  **TextBox**.

Служить для того, щоб користувач міг ввести текст під час роботи програми.

Властивості:

Text — містить символи, які ввів користувач. Інші властивості аналогічні елементу «Напис».

Рамка  **Frame** — використовується для оформлення, а також для угрупування перемикачів. Зверху на рамці можна зробити напис за допомогою властивості **Caption**. Якщо потрібно створити елемент усередині рамки, то перед їхнім створенням рамку виділяють, тоді рамка може служити контейнером для групи перемикачів.

Кнопка  **Command Button.**

Властивості:

– **Caption** — напис на кнопці.

– **Enabled** — доступність елемента. За допомогою цієї властивості блокуються елементи, які користувачеві не можна використовувати в цей момент. Заблоковані елементи відображаються сірим кольором. У заблоковане текстове поле не вийде ввести текст, а заблоковану кнопку не можна натиснути.

Вибирають із двох значень:

- **True** (Так) — елемент управління доступний користувачеві;
- **False** (Немає) — елемент управління не доступний;

– **Visible** — видимість елемента управління:

- **True** (Так) — видний;
- **False** (Немає) — не видний.

Прапорець  — **Check Box.**

Використовується, коли користувач повинен ввести **Так** (прапорець установлений) або **Ні** (прапорець знятий).

Властивості:

Value — містить значення елемента управління. Є такі значення:

0 — ні, прапорець знятий;

1 — є, прапорець установлений;

2 — прапорець недоступний.

Перемикач  **— Option Button.**

Дозволяє користувачеві вибрати один варіант із декількох.

Властивості

Value — показує обрану опцію **Так** (1) або **Ні** (0).

Поле зі списком  **— ComboBox.**

У це поле користувач може вводити текст так само, як і в **TextBox**, а крім того, якщо натиснути  , то відкриється список, з якого можна вибрати потрібний рядок.

Властивості:

– **Text** — вміст рядка, уведений користувачем або обраний зі списку;

– **List** — рядок списку (багаторядкова властивість);

– **ListIndex** — номер обраного користувачем рядка (нумерація починається з нуля, якщо жодний рядок не був обраний, то властивість дорівнює — 1).

Список **ListBox** 

Містить список рядків, у якому користувач може вибрати один або кілька рядків.

Властивості елемента аналогічні елементу **ComboBox**, за винятком властивості **Text**, якого тут немає.

Рамка для рисунка **PictureBox**  **.**

Містить рисунок. Він вибирається за допомогою властивості **Picture**, у яке вводиться ім’я файлу або вибирається за допомогою кнопки  .

4 Створення програмного коду

4.1 Поняття програмного коду

Для того щоб програма виконувала запропоновані їй дії, наприклад обчислювала, виводила результат, реагувала на дії користувача, наприклад на натискання кнопок, вибір рядків зі списку, необхідний програмний код.

Програмний код — це набір слів і символів мови програмування.

Слова та символи мають бути записані суворо за правилами мови, без орфографічних і пунктуаційних помилок. Найточніший запис дозволить комп'ютеру однозначно зрозуміти та виконати програму.

4.2 Вікно програмного коду

Програмний код записується у вікні коду. Таке вікно є у кожної форми. Структуру вікна коду подано на рисунку 10.

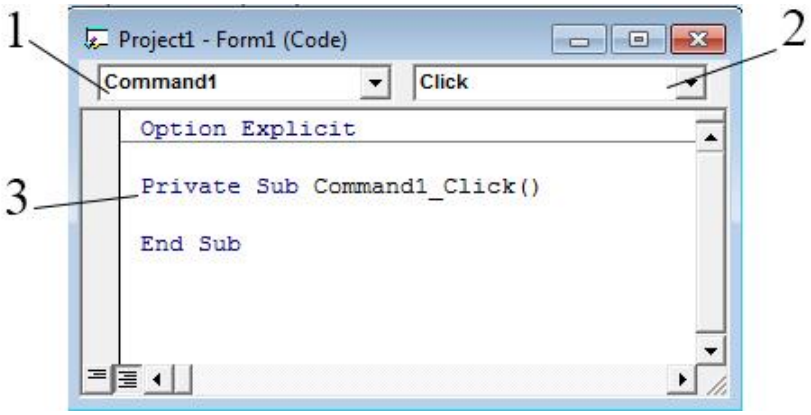
Відкрити вікно коду:

1 спосіб — у вікні Провідника проекту натиснути правою кнопкою на потрібній формі й у меню, що відкрилося, вибрати **View Code** (Показати код).

Вікно коду може бути й не пов'язане з формою. Окреме вікно коду називається **Модуль**  . Модулі у вікні Провідник проекту згруповані в групу **Модулі**. Для відкриття вікна з кодом модуля потрібно у вікні Провідник проекту двічі натиснути по імені модуля.

2 спосіб — двічі натиснути по елементу управління на формі або по самій формі у вікні форми.

Примітка – При цьому не тільки відкривається вікно коду, але й створюється процедура обробки події.



1 — список елементів управління;
2 — список подій елементів управління;
3 — процедура (код)

Рисунок 10 — Структура вікна коду

4.3 Процедури

Процедура — це відособлений фрагмент програмного коду, за допомогою якого вирішується звичайно невелике завдання.

Процедури бувають:

– **Процедури обробки подій.** Виконуються при виникненні якої-небудь події в якому–небудь елементі управління (або формі).

– **Довільні процедури.** Вони не пов’язані з подіями та можуть бути викликані з будь-якої іншої процедури, виконані в будь-який час.

Структура процедури

Процедура складається з таких елементів:

– **Заголовок процедури** — зазначає початок процедури, її тип, призначення (подія).

– **Закінчення процедури** — закінчує програмний код процедури.

– **Тіло процедури** — це рядки між заголовком і закінченням. Їхня кількість необмежена. Рядки містять приписи, які мають виконатися при виклику процедури (виникненні події).

Створення процедури

Для створення процедури необхідно виконати такі дії:

1-й спосіб — двічі натиснути по потрібному елементу управління або формі. Відкриється вікно коду, а в ньому з'явиться заголовок і закінчення процедури.

Якщо необхідна інша подія, то її обирають за допомогою списку у верхньому правому куті вікна коду.

2-й спосіб — відкрити вікно коду.

Tools (Інструменти**) => Add Procedure (**Додати процедуру**) =>** укажіть ім'я й параметри процедури **=> Ok**.

3-й спосіб — відкрити вікно коду та ввести потрібні рядки із клавіатури.

Виклик процедур на виконання

Щоб виконалася процедура обробки події, ця подія повинна відбутися.

Для виконання довільної процедури в тілі іншої процедури вказують ім'я цієї процедури.

```
Private Sub Command1_Click( )
Kvadrat
End Sub
```

Тут при натисканні на кнопку **Command1** виникає подія **Click** (натиснення мишею) і викликається та виконується процедура **Kvadrat**.

Код процедури виконується за порядком та зверху униз.

5 Змінна

5.1 Поняття змінної

Змінна — це пойменована комірка пам'яті, що зберігає яке-небудь одне значення (одне число, один фрагмент тексту).

Змінна має **ім'я** та **значення**.

Змінні служать для зберігання вихідних даних, використовуваних у програмі, а також результатів обчислень.

Використовуючи змінні, можна скласти програму «у загальному вигляді», і програма буде виконуватися при будь-яких припустимих вихідних даних.

Примітка – Властивості об'єктів, по суті, також є змінними, тому що теж зберігають певні числові або текстові значення.

5.2 Ім'я змінної

Ім'я змінної — це рядок символів, що відрізняє її від інших змінних і об'єктів програми (елементів управління).

Таким чином, імена змінних мають бути унікальними.

Правила написання імен змінних:

1 Ім'я змінної має починатися з ***букви.***

2 Іншими символами можуть бути ***букви*** (рядкові або прописні), ***цифри*** та ***символ підкреслення. Пробіл, крапка, кома й інші спеціальні знаки*** – неприпустимі.

3 Довжина імені не має перевищувати *255 символів*.

4 Ім'я змінної не повинне збігатися із ключовими словами VB (наприклад: If, Then, For, To, Next, Print і іншими).

5.3 Значення змінної

Значення змінної — це дані, які в ній зберігаються.

Тип даних (тип змінної) визначає те, як зберігаються та обробляються дані. Основні типи змінних наведені у таблиці 2.

Наприклад: 2 + 3 = 5 — числа, а "2" + "3" = "23" — текст.

Таблиця 2 — Основні типи змінних

Тип даних	Об’єм пам’яті (байт)	Діапазон значень	Префікс	Суфікс	Приклад
1	2	3	4	5	6
Цілі числа					
Byte (однобайтне ціле число)	**1**	**Додатне число від 0 до 255**	**byt**		**bytImage**
Integer (коротке ціле число)	**2**	**Від -32768 до 32767**	**int**	**%**	**intQuantity**
Long (довге ціле число)	**4**	**Від -2147483648 до 2 147483648**	**lng**	**&**	**lngTotal**
Boolean (логічне значення)	**2**	**True (**ІСТИНА**) або False (**ХИБНІСТЬ**)**	**bln**		**blnSuccess**
Числа з плаваючою крапкою					
Single (дійсне число із плаваючою крапкою (нормальне, одинарне))	**4**	**Від’ємні числа: від -3.4E+38 до -1.4E-45 Додатні числа: від 1.4E-45 до 3.4E+38**	**sng**	**!**	**sngLength**
Double (дійсне число із плаваючою комою подвійної точності (довге))	**8**	**Від’ємні числа: від -1.8D+308 до -4.9D-324 Додатні числа: від 4.9D-324 до 1.8D+308**	**dbl**	**#**	**dblSum**
String (рядок змінної довжини)	**10 байтів + довжина рядка (1 байт на кожний символ)**	**від 0 до 2 мільярдів символів**	**str**	**$**	**strLastname**
String *довжина (рядок фіксованої (постійної) довжини))	**довжина рядка (1 байт на кожний символ**	**від 1 до ~65400**		**$**	

Продовження таблиці 2

1	2	3	4	5	6
Об'єктні типи					
Object (посилання на об'єкт)	**4**				
Не визначені типи					
Variant (числові типи)	**16**	**довільне числове значення**	**vnt**		**vntValue**
Variant (символьні типи)	**22 байти+ довжина рядка**	**довільне символьне значення**	**vnt**		
Інші типи					
Currency (грошова величина-число з фіксованою крапкою)	**8**	**Ціла частина числа до 15 цифр, дробова – до 4 Від -922337203685477. 5808 до922337203685477.5807**	**cur**	**@**	**curPrice**
Date (дата/час)	**8**	**Діапазон дат від 1 січня 100 року до 31 грудня 9999 року. Діапазон часу від 00:00:00 до 23:59:59.**	**Dtm**		**dtmFinish**

Змінна типу **Variant** може приймати будь–який тип залежно від значення, що зберігається в ній, однак, займає більше пам'яті.

Якщо змінна не була оголошена, то вона має тип Variant.

5.4 Присвоєння значення змінній

Для присвоєння значення служить **оператор присвоювання,** яким є знак дорівнює (=).

Оператор — це слово або знак, що виконує одне приписання (одну операцію).

Ліворуч від знака «дорівнює» вказується **ім'я змінної**, у яку буде поміщене значення, а **праворуч** — **значення змінної** (числове або текстове) або **математичний**, або **символьний вираз**, або **інша змінна**, з якої береться значення. Тобто загальний вигляд оператора присвоювання:

Куди (змінна) = **що** або **звідки**

У математичних виразах використовуються знаки арифметичних операцій (рисунок 11).

^	піднесення у степінь
*	множення
/	ділення
+	додавання
-	віднімання

Рисунок 11 — Знаки арифметичних дій

Арифметичні операції виконуються в такій послідовності: спочатку піднесення у степінь, потім множення та ділення, в останню чергу — додавання та віднімання. Якщо у виразі кілька операцій, то вони виконуються зліва направо.

Для зміни порядку операцій використовуються тільки круглі дужки ( ).

Математичний вираз спочатку обробляється та обчислюється результат, що потім присвоюється змінній (міститься в змінну).

При написанні чисел у коді програми ціла та дробова частина числа відділяються крапкою.

При виконанні такого рядка спочатку обчислюється права частина, потім результат присвоюється змінні, що розташована ліворуч від знака «дорівнює».

Аналогічно присвоюються **значення властивостям**. Це записується так:

Об’єкт.Властивість = Значення

Примітка – Оскільки VB може сам визначити тип змінної за її значенням (див. Variant), то тип змінної можна не вказувати. Однак оголошення змінних є правилом гарного тону програмування, а також дозволяє уникнути деяких помилок у

програмі, таких як неоднакове написання імені змінної в різних місцях програми, недотримання типів даних в обчисленнях.

Змінну оголошують на початку вікна коду або на початку процедури за допомогою оператора **Dim**:

Dim Ім’я_змінної **As** Тип_змінної

Як Тип змінної вказуються слова Byte, Long, String і інші з таблиці типів (таблиця 2).

Наприклад: Dim a As Byte

При оголошенні декількох змінних можна перелічувати їх через кому:

Dim a As Byte, b As String.

6 Умовні оператори

Для реалізації алгоритму з розгалуженням необхідний умовний оператор, який має дві основні форми:

Лінійна форма

IF *умова* **THEN** *блок_операторів-1* [**ELSE** *блок_ операторів*-2]

Примітка – Доцільно використовувати, якщо перевіряється одна умова.

Блокова форма

IF <*умова1*> **THEN**
[*блок_операторів-1*]
[**ELSEIF** < *умова 2*> **THEN**
[*блок_ операторів*-2]].
...
[**ELSE**
[*блок операторів-n*]]
END IF

Примітка — Квадратні дужки не ставляться, а лише показують те, що вміст, що перебуває між ними, можна опустити у випадку непотрібності.

Працює цей оператор у такий спосіб:

1 Перевіряється *умова*.

2 Якщо умова істинна, то виконується *блок операторів* після **IF** або **ELSEIF.**

3 Якщо умова хибна, то виконується *блок операторів* після **ELSE.**

Як умова може бути використаний будь-який логічний вираз.

Наприклад:

```
If x>0 Then
        MsgBox "Число додатне"
Else
        MsgBox "Число від’ємне"
End If
```

7 Оператор циклу

Загальний вигляд оператора:

For ***Лічильник*** **=** ***Початкове значення*** **To** ***Кінцеве значення*** **[Step** ***Крок*****]**

Оператори тіла циклу **(***виконуються кілька разів*)

Next ***Лічильник***

Умовні позначки:

1 ***Лічильник*** — змінна, яка містить поточне значення циклу. При кожному обороті циклу ***Лічильник*** змінюється на ***Крок***.

2 ***Початкове значення***, ***Кінцеве значення***, ***Крок*** — числа, вирази, змінні або властивості об’єктів, що містять відповідні числові значення.

3 ***Лічильник*** починає відлік від ***Початкового значення***, але не може перевищити ***Кінцевого значення***.

Примітка – ***Крок*** можна опустити, тоді він буде дорівнювати 1.

Крок може бути < 0, тоді значення ***Лічильника*** буде убувати, а ***Початкове значення*** має бути більше ***Кінцевого значення.***

Робота оператора:

1) обчислюються значення арифметичних виразів (***Початкове значення***, ***Кінцеве значення***, ***Крок***);

2) ***Лічильнику*** присвоюється ***Початкове значення*** (формується лічильник циклу);

3) перевіряється умова: якщо ***Лічильник*** > 0 і ***Лічильник*** ≤ ***Кінцеве значення*** або ***Крок*** < 0 і ***Лічильник*** ≥ ***Кінцеве значення*** то виконується ***Оператори тіла циклу***; значення ***Лічильника*** змінюється на величину ***Кроку*** і повторюється весь пункт. У протилежному випадку виконуються програмні рядки, що розташовані після **NEXT**.

Приклад:

```
For i=1 To 10 Step  2
    Print i
Next i
```

Такий цикл повториться 5 разів. Змінна і буде приймати значення від 1 до 10 із кроком 2, тобто 1, 3, 5, 7, 9. Ці числа будуть надруковані на формі за допомогою оператора **Print**.

8 Вбудовані функції

8.1 Поняття функції

Функція обчислює та повертає результат залежно від вихідних даних (аргументів).

Наприклад: sin(*x*)

Тут *x* – аргумент, а обчислений синус від *x* - результат.

Загальний вигляд функції:

Ім’я функції (*аргумент_1, аргумент_2, аргумент_n*)

Імена функцій складаються за тими ж правилам, що й імена змінних.

Функція може мати один або кілька аргументів.

Функції можуть використовуватися в арифметичних виразах з оператором присвоювання, наприклад:

y = sin(x)

y = 2 * sin(4 * x) + 2

або з оператором порівняння:

sin(x) > cos(y) + 1

sin(Pi) = 0.

У VB передбачені **вбудовані функції**, такі як sin, cos, які обчислюються певними правилами, і також функції може створювати програміст для вирішення спеціальних завдань.

Основні групи вбудованих функцій:

– математичні функції;
– символьні функції;
– функції для роботи з датою та часом;
– функції для перетворення типів даних.

8.2 Математичні функції

Математичні функції призначені для роботи із числовими даними, що є аргументами функцій. Найбільш поширені математичні функції наведені в таблиці 3.

Таблиця 3 — Перелік математичних функцій

Запис на VB	***Функція***	***Приклад***
1	2	3
SIN(X)	синус X	**sinX**, аргумент у радіанах
COS(X)	косинус X	**CosX**, аргумент у радіанах
TAN(X)	тангенс X	**tgX**, аргумент у радіанах
ATN(X)	арктангенс X	**ArctgX**, $-\pi/2<X<\pi/2$,
ABS(X)	модуль X	**|X|**

Продовження таблиці 3

1	2	3
SQR(X)	квадратний корінь X	**√X**, X>0
EXP(X)	експонента X	$\mathbf{e^X}$
LOG(X)	натуральний логарифм X	**lnX, X>0**
FIX(X)	усікання до цілого (відкидання дробової частини)	FIX(-5.6)=-5 FIX(24.07)=24.00
INT(X)	знаходження найбільшого цілого, що не перевищує X	INT(15.5)=15 INT(-6.2)=-7 INT(-6.7)=-7
RND(X)	Генерація псевдовипадкових чисел від 0 до 1	
ROUND	округлення до заданого числа десяткових знаків	Round(3.44,1)=3.4 Round(3.44)=3.4
SGN(X)	знак числа X	SGN(X) = -1, якщо X < 0 SGN(0) = 0, якщо X = 0 SGN(X) = +1, якщо X > 0

8.3 Символьні функції

Рядок (текст) — це послідовність будь–яких символів.

Рядки в тексті програми записують у подвійних лапках (але не імена змінних, що їх містять).

Наприклад:

a = "Вася"

b = a + " Іванов".

Примітка — Тут показаний пробіл ␣ *перед* Іванов.

Рядок може бути порожнім, тобто не містити жодного символа.

Для позначення порожнього рядка лапки пишуть без пробілів між ними: a = "". Основні символьні функції вказані у таблиці 4.

Таблиця 4 — Перелік основних символьних функцій

Функція	***Призначення***
1	2
Asc	Повертає ASCII-Код символу
Chr	Перетворює ASCII-Код у символ
InStr, InStrRev	Здійснюють пошук одного рядка в іншому
LCase	Змінює регістр букв вихідного рядка на нижній
Left	Повертає зазначену кількість символів з початку рядка
Len	Повертає кількість символів у рядку
LTrim, RTrim, Trim	Видаляють пробіли, розташовані відповідно на початку, наприкінці та по обидва боки символьного рядка
Mid	Повертає задану кількість символів з довільного місця рядка
Right	Повертає зазначену кількість символів з кінця рядка
Str, CStr	Перетворюють числовий вираз у рядок
StrReverse	Змінює порядок проходження символів у рядку на зворотний

Продовження таблиці 4

1	2
StrConv	Змінює регістр букв символьного рядка
Val	Перетворюють рядок у числовий вираз
UCase	Змінює регістр букв вихідного рядка на верхній

8.4 Функції для роботи з датою та часом

Date() — функція не має аргументів і повертає поточну дату.

Time() — функція також не має аргументів і повертає поточний час.

8.5 Функції перетворення типів даних

Якщо необхідно, щоб результатом виразу було число, то й всі вихідні дані в цьому виразі мають бути числами.

І так само, якщо результат — рядок, то всі вихідні дані мають бути рядками.

У протилежному випадку можливі або помилки в програмі, або результат не буде відповідати очікуваному. Приклад використання функцій перетворення типів даних указаний у таблиці 5.

Таблиця 5 — Функції перетворення типів даних

Функція	**Опис**	**Приклад**	
		Аргументи	***Результат***
Val (рядок)	Перетворює рядок у число. Якщо у рядку немає цифр, то результат буде 0	Val(“25”) Val(“Мир”) Val(“01рахунок”)	25 0 1
Str (число)	Перетворює число в рядок	Str(5) Str(-5)	5 -5

8.6 Форматування чисел

Функція ***Format*** (таблиця 6) перетворює числові значення в текстовий рядок і надає можливість управляти зовнішнім виглядом (появою) рядка.

Таблиця 6 — Символи, які використовуються функцією ***Format***

Символ	***Опис***
0	Цифровий символ-заповнювач; друкує замикаючий або провідний нуль у поточній позиції
#	Цифровий символ-заповнювач; ніколи не друкує замикаючих або провідних нулів
.	Символ–Заповнювач десяткової точки
/	Символ–Заповнювач для роздільника тисяч
- +$( )space	Буквальний символ; символи відображаються точно так, як вони набрані у форматному рядку

Наприклад:
Format(8315.4, "00000.00")= 08315.40
Format(8315.4, "#####.##") =8315.4.

Список літератури

1 Бутенко, В. М. Основи програмування мовами високого рівня для студентів вищих навчальних закладів [Текст]: навч. посібник / В.М. Бутенко, В.С. Меркулов, О.В. Казанко, О.В. Чаленко. – Харків: УкрДАЗТ, 2009. —206 с.

2 Данько, М.І. Математичні методи та моделі в розрахунках на ЕОМ з дисципліни «Математичні моделі на ЕОМ» для студентів всіх спеціальностей всіх форм навчання [Текст]: навч. посібник / М.І. Данько, В.С. Меркулов, В.О. Гончаров, І.Г. Бізюк, В.М. Бутенко, О.В. Головко. — Харків: УкрДАЗТ, 2008. —174 с.

3 Меркулов, В.С. Основи алгоритмізації базових обчислювальних процесів з дисципліни «Обчислювальна техніка, програмування, моделювання систем» для студентів всіх спеціальностей всіх форм навчання [Текст]: навч. посібник / В.С. Меркулов, В.О. Гончаров, І.Г. Бізюк, В.М. Бутенко, О.В. Головко. – Харків: УкрДАЗТ, 2008 —164 с.

4 Філіппенко, І.Г. Конспект лекцій з дисципліни “Математичні методи і моделі в розрахунках на ЕОМ”. Основи проектування технічних систем. Класифікація моделей. Основи теорії оптимізації [Текст] / І.Г. Філіппенко, В.О. Гончаров, В.С. Меркулов. – 2-ге вид., випр. - .Харків, УкрДАЗТ, 2004. — Ч.2. – 47 с.

5 Філіппенко, І.Г. Конспект лекцій з дисципліни “Математичні методи і моделі в розрахунках на ЕОМ”. Методи оптимізації [Текст] / І.Г. Філіппенко, В.О. Гончаров, В.С. Меркулов. – 2-ге вид., випр. – Харків: УкрДАЗТ, 2004. – Ч.2. – 38 с.

6 Браун, С. Visual Basic 6.0. [Текст]: учебный курс / С. Браун.– СПб.: Питер, 2006.–574 с.

7 Волченков, Н. Г. Программирование на Visual Basic 6.0 [Текст]: в 3-х ч. / Н. Г. Волченков. – М: ИНФРА-М, 2000. – 280 с.

8 Гридюшко, В.И. Вагонное хазяйство [Текст]: учеб. пособие для вузов / В.И. Гридюшко, В.П. Бугаев, Н.З. Криворучко. – М.: Транспорт, 1988. – 295 с.

Додаток А

Приклад виконання розрахункової частини курсової роботи

У розрахунковій частині курсової роботи необхідно обрати числовий метод, розробити алгоритм, скласти та налагодити відповідну програму алгоритмічною мовою високого рівня, отримати результати та зробити висновки.

А.1 Постановка та математичне формулювання задачі

Механізовані пункти підготовки вагонів (МППВ) до перевезень створюються на великих навантажувальних відділеннях ряду залізниць, де накопичується велика кількість несправних вагонів, що прибувають під вивантаження за регулювальними завданнями [8].

Так, на станції Кемеровської залізниці створений найбільший МППВ для середньодобової підготовки до перевезень 1500 чотиривісних напіввагонів. Критий ангар пункту розрахований на одночасне обслуговування та поточний ремонт трьох складів по 60 чотиривісних напіввагонів у кожному. Ангар розділений на шість ділянок, оснащених засобами механізації для одночасного ремонту кузовів, рам, ходових частин, автогальм і автозчепного обладнання. Перед постановкою в ангар напіввагони очищають від залишків вантажів, що перевозяться.

На Донецькій залізниці робота механізованих пунктів, створених на станціях Штерівка і Довжанська, побудована за іншим принципом. Напіввагони складами подають на колії поточного безвідчіпного ремонту. Ці колії обладнані самохідними ремонтними установками типу РУ. Напіввагони, що вимагають великого обсягу ремонту кузовів, подають у криті приміщення. Вони обладнані вагоноремонтними машинами «Донбас» і іншими засобами механізації, що дають змогу виконувати поточний ремонт вагонів у повному обсязі. Для заміни колісних пар або ремонту напіввагона з попереднім

підніманням кузова передбачена окрема позиція, обладнана піднімальними засобами, запасом справних колісних пар, електрогазозварювальним та іншими пристроями. Пропускна спроможність вказаних механізованих пунктів досягає 1000 чотиривісних вагонів на добу.

Маючи у своєму розпорядженні необхідні технічні засоби для поточного ремонту вагонів, МППВ забезпечують високий рівень відновлення працездатності вагонів. Використання засобів механізації ремонтних процесів дає можливість на кожному вагоні в середньому освоювати обсяг ремонту, у 1,5—2 разу більший, ніж при виконанні ремонтних операцій ручним способом. Велику кількість несправностей без застосування вагоноремонтних машин не може бути усунено взагалі до надходження вагонів у деповський або капітальний ремонт.

Організація роботи і технічне оснащення МППВ залежать від програми пункту, типу ремонтованих вагонів, типу і розташування станції, на якій він розміщується, наявності (або відсутності) на цій станції вагонного депо і багатьох інших чинників. Вибір основних технічних засобів МППВ залежить також від того, чи будуть вагони, що вимагають поточного відчіпного ремонту, подавати на спеціально виділені колії або їх працездатність відновлюватиметься без відчеплення від составів.

На МППВ з відносно невеликою програмою (200—400 вагонів на добу) може застосовуватися один з двох варіантів організації поточного відчіпного ремонту: на коліях безвідчіпного ремонту з використанням вагоноремонтних машин портального типу, виготовлених за проектом Т-337 ПКБ ЦВ, або на спеціально виділених відкритих коліях, обладнаних машинами типу «Донбас». В останньому варіанті для поточного відчіпного ремонту в холодних кліматичних зонах будують спеціальні криті приміщення (ангари).

Фронт поточного відчіпного ремонту вагонів (у критих ангарах) розраховується за формулою

$$\Phi = \frac{N_o \left(t_o + t_y \right)}{m_п T_a}, \qquad \text{(А.1)}$$

де N_o — найбільше добове надходження вагонів у поточний відчіпний ремонт (середнє з найбільших місячних значень за рік);

t_o — тривалість поточного відчіпного ремонту, год; $t_o \approx 3{,}5$ год;

t_y — час, необхідний на прибирання відремонтованих і подавання несправних вагонів, год;

$m_п$ — число колій в ангарі;

T_a — встановлена тривалість роботи ангара на добу, год.

Довжина ангара

$$L_a = \Phi \cdot l_в + (\Phi - 1) \cdot l_i + 2 \cdot a, \qquad (А.2)$$

де $l_в$ — довжина вагона за осями зчеплення автозчепів, м;

l_i — інтервал між сусідніми вагонами, l_i = 1,5 ÷ 2,0 м;

a — відстань від торцевої стіни ангара до автозчепу крайнього вагона, a = 3,5 ÷ 4 м.

Якщо вагонне депо віддалене від пункту підготовки вагонів до перевезень, на території останнього передбачають ремонтні майстерні, що мають ковальське, електрозварювання, механічне, столярне відділення, компресорну, інструментальне відділення.

Шляхи поточного безвідчіпного ремонту вагонів обладнують самохідними ремонтними установками, лініями електрозварювань, пристроями для централізованого огородження поїздів і випробування гальм, гучномовним зв'язком, засобами для подовжнього і поперечного транспортування запасних частин і матеріалів, механізмами та пристроями для ремонту вагонів. На території МППВ розміщують службово-побутові і виробничі приміщення. Зовнішнє освітлення паркових шляхів має відповідати встановленим нормам.

На великих вантажних відділеннях є особливості в організації підготовки вагонів до перевезень. За існуючою технологією вагони, що направляються в райони завантаження за регулювальними завданнями, проходять технічний контроль безпосередньо на станціях завантаження. Вагони з несправними

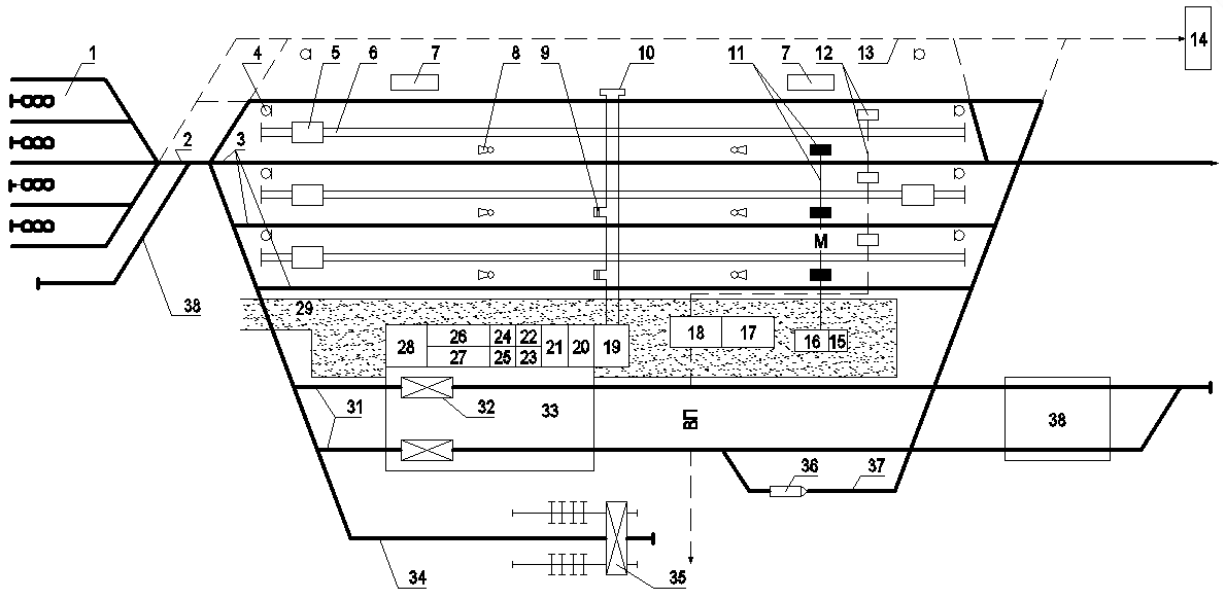
кузовами, що вимагають відчіпного ремонту, як правило, разом із справними прямують до цих станцій. Тут здійснюється великий обсяг маневрової роботи з відчеплення несправних вагонів. Це призводить до затримки подачі справного рухомого складу на станції навантаження і тривалих простоїв вагонів, що чекають пересилки в ремонтні пункти часто на значну відстань.

Найбільш досконалу форму організації технічного обслуговування і підготовки до перевезень на великих вантажних дорогах і відділеннях, що одержують велику кількість порожніх вагонів за регулювальними завданнями, має система централізованого відбору несправних вагонів на одній або декількох станціях і організованого напряму їх у ремонт на механізовані пункти. В цьому випадку на вантажні станції надходитиме тільки справний рухомий склад.

Централізований відбір несправних вагонів з порожніх вагонопотоків доцільно проводити за нижченаведеною технологією.

Порожні состави, що прибувають у парк приймання, оглядають і відмічають вагони, що вимагають поточного відчіпного ремонту. Справні вагони проходять технічне обслуговування ходових частин і підготовку кузовів до перевезення вантажів на місці, тобто на коліях парку приймання, які оснащені самохідними ремонтними установками. Потім прибулі вагони сортують за допомогою гірки або напівгірки. Несправні вагони відповідно до розмітки розпускають на підгіркові колії, а потім після накопичення до повних складів подають на МППВ. Очищають вагони від залишків вантажів, що перевозяться, на спеціально виділених коліях. Справні вагони, сформовані в состави, направляють безпосередньо в райони завантаження.

Типову схему об’єкта розрахунку (МППВ) вантажних вагонів до перевезень зображено на рисунку А.1.



1 – приймальний парк; 2 – напівгірка; 3 – колії технічного обслуговування вагонів; 4 – сигнали огородження; 5 – ремонтна установка; 6 – колії руху ремонтних установок; 7 – приміщення для обігріву ремонтно-оглядових бригад; 8 – переговірна колонка; 9 – ходовий люк; 10 – тунель; 11 – позиції запасних частин; 12 – повітропровід (ВП) з колонками централізованого випробування гальм; 13 – колії для сумісного використання пунктом і станцією; 14 – пристрій очищення від залишків вантажу; 15, 16 – допоміжні приміщення; 17 – виробничо-побутові приміщення; 18 – приміщення оператора пункту; 19 – комора запасних частин; 20 – відділення ремонту обладнання та оснащення пункту; 21 – слюсарно-механічне відділення; 22 – ковальське відділення; 23 – електрозварювальне відділення; 24 – відділення зарядки акумуляторних батарей; 25 – інструментальна комора; 26 – відділення ремонту окремих частин рухомого складу; 27 – деревообробне відділення; 28 – гараж; 29 – асфальт; 30 – витяжка; 31 – колії ТОв–1; 32 – універсальна самохідна правильна машина; 33 – ангар; 34 – колії завантаження-вивантаження колісних пар; 35 – козловий кран; 36 – мотовоз; 37 – колії стоянки мотовоза; 38 – позиція розділення вагонів у металобрухт

Рисунок А.1 – Схема МППВ

А.2 Алгоритм і його опис

При створенні алгоритму (рисунок А.2) використовується ряд змінних, ідентифікатори яких зведені у таблиці А.1.

Таблиця А.1 — Таблиця ідентифікаторів

Назва ідентифікатора	Вміст ідентифікатора	Тип ідентифікатора
Вхідні величини:		
N	найбільше добове надходження вагонів, ваг	Integer
t	тривалість технічного обслуговування з відчепленням, год	Single
tu	час на прибирання та подачу вагонів, год	Single
m	кількість колій в ангарі, шт	Integer
Ta	тривалість роботи ангара на добу, год	Single
La	довжина вагона за осями зчеплення автозчепів, м	Single
lv	інтервал між сусідніми вагонами, м	Single
a	відстань від торцевої стіни ангара до автозчепу крайнього вагона, м	Single
Розрахункові дані:		
F	фронт технічного обслуговування з відчепленням	Integer
Lan	довжина ангара, м	Single

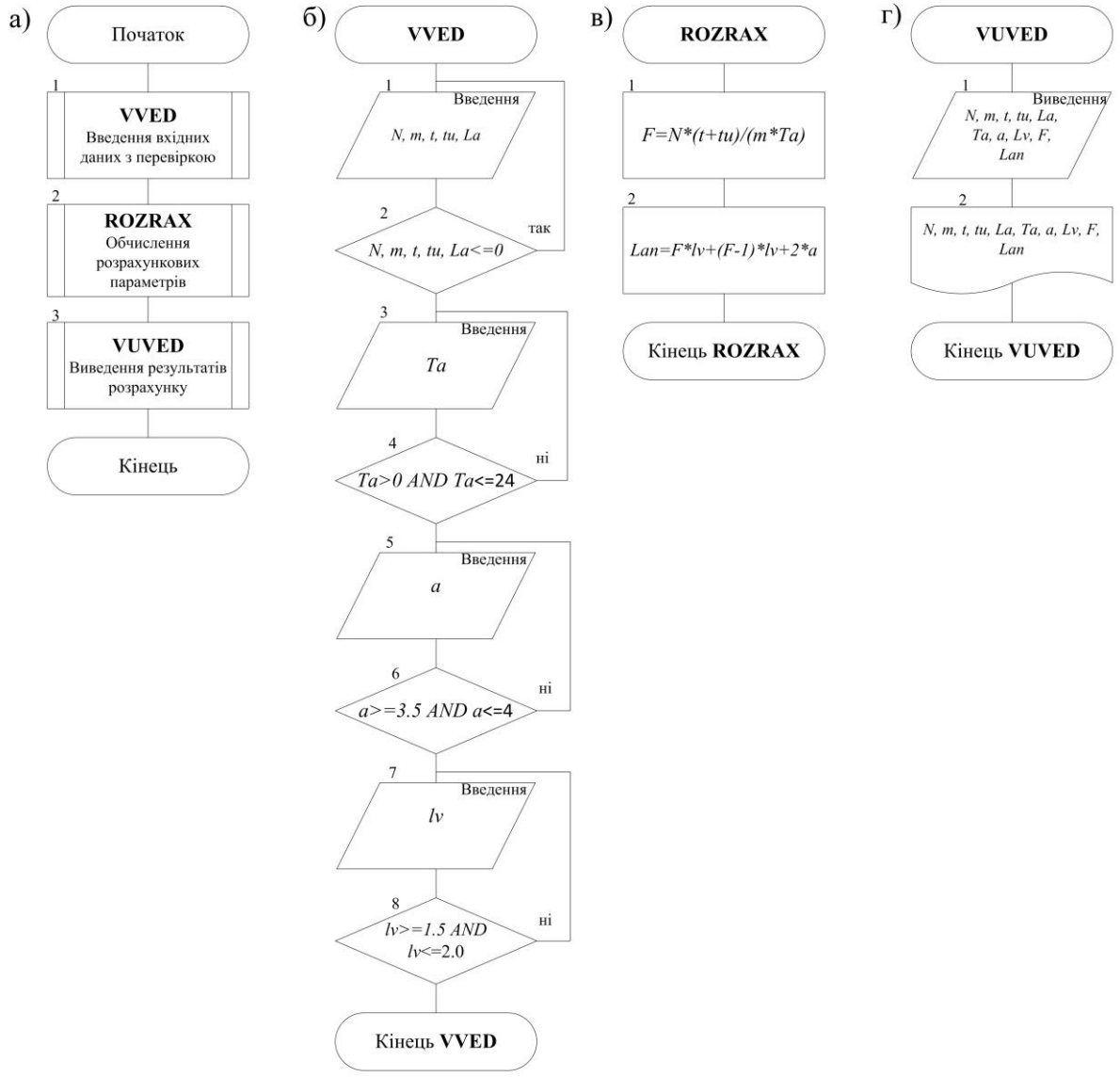


Рисунок А.2 — Алгоритм розрахунку

На рисунку А.2 блоки — початок (кінець) відповідно показують початок та кінець дій, що виконуються в алгоритмі в цілому та окремих процедурах.

На рисунку А.2, а зображено загальний алгоритм розв’язання задачі, на якому зображено:

процедура VVED — введення вхідних даних з перевіркою;

процедура ROZRAX — обчислення розрахункових проміжних та підсумкових змінних;

процедура VUVED — виведення результатів розрахунків.

На рисунку А.2, б зображено алгоритм процедури VVED.

1 — задання значень даних, таких як:

– найбільше добове подавання вагонів у поточний відчіпний ремонт, *N*;

– число колій в ангарі, *m*;

– довжина вагона за осями зчеплення, *la*;

– тривалість технічного обслуговування, *t*;

– час на прибирання і подавання вагонів, *tu*;

2 — перевірка введених значень;

3 — введення значення тривалості роботи МППВ, *Ta*;

4 — перевірка правильності введення значень;

5 — введення відстані від торцевої стіни до автозчепу крайнього вагона, *a*;

6 — перевірка правильності введення значення;

7 — введення значення відстані між сусідніми вагонам, *lv*;

8 — перевірка правильності введення значення відстані.

На рисунку А.2, в зображено алгоритм процедури ROZRAX.

1 — розрахунок фронту ТОв-1;

2 — розрахунок довжини ангара для проведення технічного обслуговування.

На рисунку А.2, г зображено алгоритм виведення введених даних та результатів розрахунку на екран (блок 1) і у формі, зручній для друку, — окремий текстовий файл (блок 2).

А.3 Текст програми

За алгоритмом складається проект (програма) розв’язання задачі.

Мовою програмування було використано Visual Basic 6.0.

```
Option Explicit

'Опис змінних
Dim N, m, F As Integer
Dim t, tu, Ta As Single
Dim lv, la, a, Lan As Single
Private Sub Form_Load()

'Опис дати
Label16.Caption = Date

End Sub
'Початок розрахунку
Private Sub Command1_Click()

'Введення даних
N = Val(Text_N.Text)
m = Val(Text_m.Text)
la = Val(Text_la.Text)
t = Val(Text_t)
tu = Val(Text_tu)

'Перевірка правильності введених даних
Do
If N <= 0 Or m <= 0 Or la <= 0 Or t <= 0 Or tu <= 0 Then
MsgBox "Неправильно введені вхідні дані. Перевірте і введіть дані повторно.", 1 + 16, "Помилка введення значень"
Exit Sub
Else
Exit Do
End If
Loop
Ta = Val(Text_Ta)
Do
'Перевірка введених даних
If Ta > 0 And Ta <= 24 Then
Text_Ta = Str(Ta)
```

```
Exit Do
Else
MsgBox "Неправильно введене значення тривалості роботи ангара на добу.", 0 + 16, "Помилка введення значень"
Ta = Val(InputBox("Введіть тривалість роботи ангара на добу.", "Введення даних", "", 4000, 500))
End If
Loop

a = Val(Text_a)
Do
'Перевірка введених даних
If a >= 3.5 And a <= 4 Then
Text_a = Str(a)
Exit Do
Else
MsgBox "Неправильно введене значення відстані від торцевої стіни до крайнього автозчепу. Обмеження (від 3,5 до 4 м). Повторіть введення", 0 + 16, "Помилка введення значень"
a = Val(InputBox("Введіть відстань від торцевої стіни ангара до крайнього автозчепу", "Введення даних", "", 4000, 500))
End If
Loop

lv = Val(Text_lv)
Do
'Перевірка введених даних
If lv >= 1.5 And lv <= 2 Then
Text_lv = Str(lv)
Exit Do
Else
MsgBox "Неправильно введене значення відстані між вагонами. Обмеження (від 1,5 до 2,0 м)", 0 + 16, "Помилка введення значень"
lv = Val(InputBox("Введіть відстань між вагонами", "Введення даних", "", 4000, 500))
End If
Loop

'Розрахунок параметрів ангара
'Фронт поточного відчіпного ремонту
F = Round(N * (t + tu) / m / Ta, 0)
'Довжина ангара для ТО
Lan = Round(F * la + (F - 1) * lv + 2 * a, 2)
'Виведення результатів розрахунку на екран
```

```
Text_F.Text = Str(F)
Text_lan.Text = Str(Lan)

End Sub
Private Sub Command3_Click()
'Виведення результатів розрахунку у файл
Open "Результати розрахунку.txt" For Append As #1
Print #1, "Дата виконання проекту "; Date
Print #1, "Вхідні дані"
Print #1, "Добове надходження вагонів у ТОв - 1 "; N
Print #1, "Число колій в ангарі "; m
Print #1, "Довжина вагона за осями зчеплення автозчепів "; la; " м"
Print #1, "Тривалість технічного обслуговування "; t; " год"
Print #1, "Час на прибирання та подавання вагонів у ТО "; tu; " год"
Print #1, "Тривалість роботи ангара на добу "; Ta; " год"
Print #1, "Відстань від торцевої стіни до крайнього автозчепу "; a; " м"
Print #1, "Відстань між вагонами "; lv; " м"
Print #1, "Розраховані значення"
Print #1, "Фронт ТОв-2 "; F; " ваг"
Print #1, "Довжина ангара "; Lan; " м"
Close #1
End Sub

Private Sub Command4_Click()
'Закінчення програми
End
End Sub
```

А.4 Аналіз отриманих результатів і висновки

Розроблена програма буде мати вигляд, зображений на рисунку А.3.

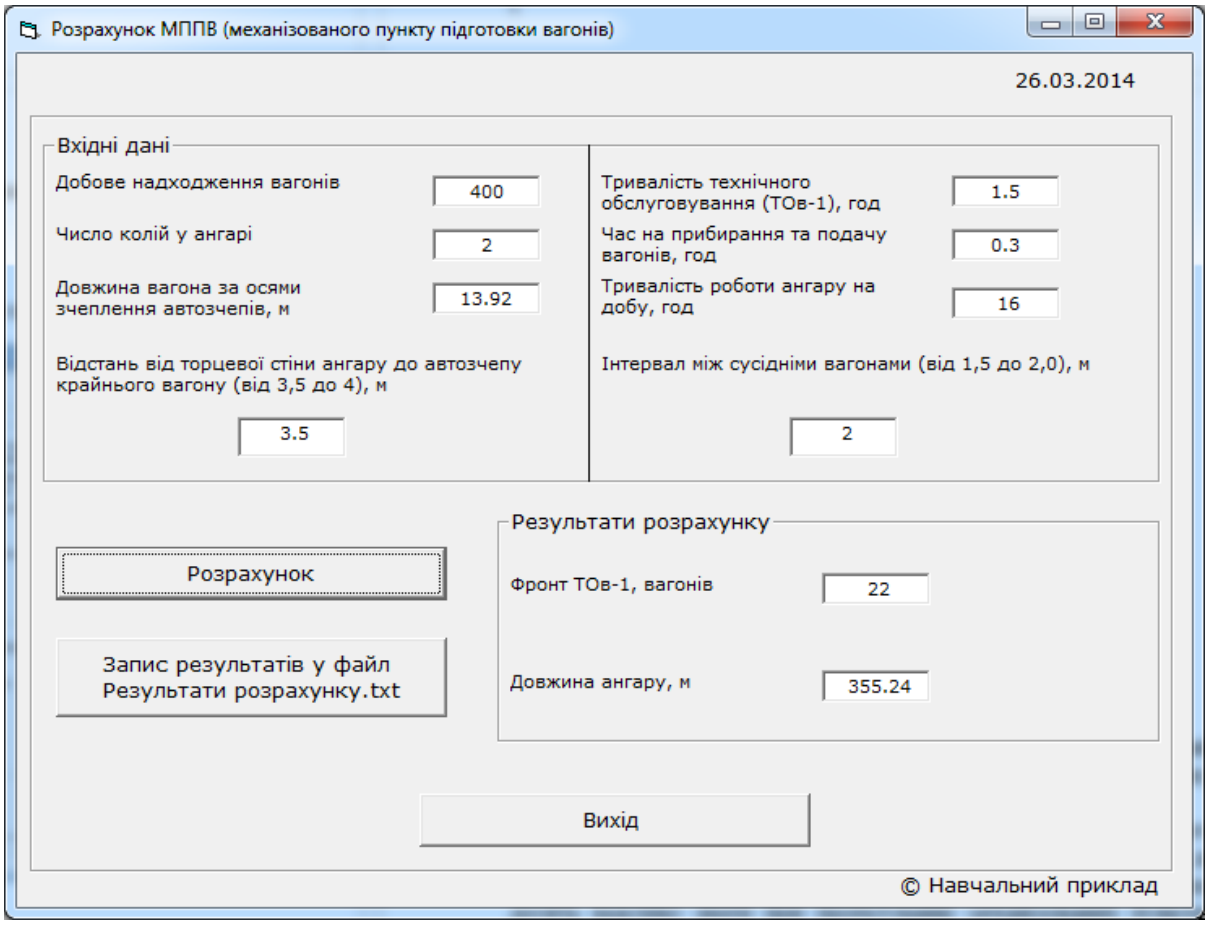


Рисунок А.3 — Приклад розробленої програми

Результати розрахунків:
Фронт ТОв-1 — 22 вагони.
Довжина ангара — 355,24 метра.

Початкові дані:
Добове надходження вагонів — 400 вагонів.
Число колій в ангарі — 2.
Довжина вагона за осями зчеплення автозчепів — 13,92 м (типовий напіввагон моделі 12-296).
Тривалість технічного обслуговування (ТОв-1) — 1,5 год.
Час на прибирання та подавання вагонів, год — 0,3 год.

Тривалість роботи ангара на добу — 16 год.
Відстань від торцевої стіни ангара до автозчепу крайнього вагона — 3,5 год.
Інтервал між сусідніми вагонами — 2 м.

Результати моделювання за необхідності зберігаються у файлі **Результати розрахунку.txt** (рисунок А.4), у каталогу розміщення програми.

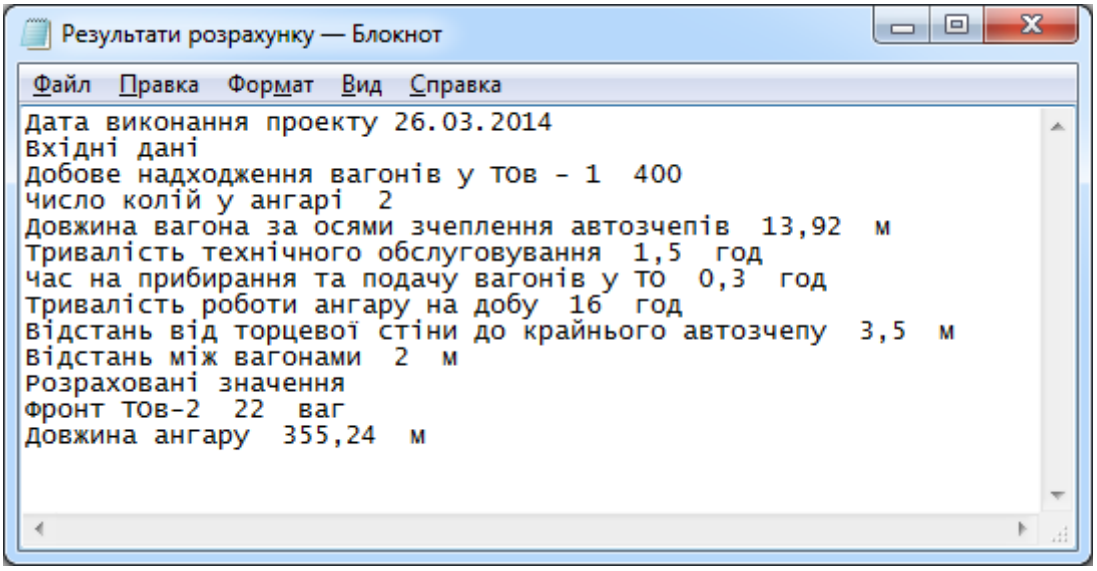


Рисунок А.4 — Результати розрахунку програми в окремому текстовому файлі

В ході виконання даної курсової роботи був складений алгоритм і розроблений проект програми розрахунку механізованого пункту підготовки вагонів до перевезень.

Складена програма дає можливість розрахувати фронт технічного обслуговування вагонів з відчепленням, які необхідно щодоби готувати до перевезень на механізованих пунктах, а також довжину ангара, що досить важливо знати при проектуванні механізованих пунктів, знаючи необхідні параметри.

За допомогою поданої програми можна провести розрахунок механізованого пункту підготовки вагонів для будь-якого великого вантажного відділення залізниці, для будь-яких видів вагонів.

Проведені на ЕОМ розрахунки відповідають реальним результатам і програмі ремонту вагонів, що діє у даний час.