МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Державний вищий навчальний заклад
“НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ”

М.О. Дудко
І.М. Мацюк
І.В. Вернер

КОМП'ЮТЕРНА ТЕХНІКА ТА ПРОГРАМУВАННЯ

Дніпропетровськ
НГУ
2010

УДК 681.3.06
ББК 32.973-01
Д 81

Рецензенти:

І.В. Жуковицький, д-р техн. наук, професор (Дніпропетровський національний університет залізничного транспорту, завідувач кафедри електронно-обчислювальних машин);

П.І. Когут, д-р техн. наук, професор (Дніпропетровський національний університет ім. О. Гончара, професор кафедри диференціальних рівнянь).

Дудко М.О.

Д 81 Комп'ютерна техніка та програмування: навч. посібник / М.О. Дудко, І.М. Мацюк, І.В. Вернер. – Д.: Вищий державний навчальний заклад "НГУ", 2010. – 140 с.

У посібнику розглянуті основні категорії апаратних і програмних засобів обчислювальної техніки. Указані базові принципи побудови архітектури обчислювальних систем. Забезпечено методичне обґрунтування процесів взаємодії інформації, даних і методів. Приведені ефективні прийоми роботи з програмним продуктом Visual Basic. Розглянуті основні засоби, прийоми і методи програмування.

Мета посібника – надання допомоги студентам при вивченні навчальної дисципліни "Комп'ютерна техніка та програмування", а також при виконанні індивідуальних завдань.

Навчальний посібник призначений для студентів технічних спеціальностей вищих навчальних закладів

УДК 681.3.06
ББК 32.973-01
© М.О. Дудко, І.М. Мацюк, І.В. Вернер, 2013
© Вищий державний навчальний заклад "НГУ", 2010

ЗМІСТ

ВСТУП

Для успішного використання комп'ютерної техніки у своїй професійній діяльності інженер повинний знати технічні можливості та специфіку програмного забезпечення електронних засобів, володіти навичками взаємодії з електронно-обчислювальною машиною (ЕОМ), але головне, він повинний уміти формулювати задачі, розробляти алгоритми їх розв'язку, записувати алгоритми мовою, яка сприймається ЕОМ.

Дисципліна “Комп'ютерна техніка та програмування” – обов'язкова в навчальних програмах більшості технічних вузів, вона покликана забезпечити фундаментальну підготовку студентів різних спеціальностей до використання комп'ютерної техніки як у навчальному процесі, так і в подальшій професійній діяльності.

Викладання дисципліни “Комп'ютерна техніка та програмування” має на меті навчити студентів:

- основним правилам експлуатації персональних комп'ютерів;
- умінню працювати з відомими операційними системами;
- умінню працювати з файловими системами;
- складати алгоритми і блок-схеми цих алгоритмів для вирішення типових практичних завдань;
- основним правилам роботи в середовищі алгоритмічних мов;
- використанню інструментальних засобів і основних операторів при розробці індивідуальних прикладних програм;
- проектуванню розгалужених і циклічних процесів в середовищі програмування мов високого рівня;
- використовувати програмні засоби алгоритмічних мов для створення графічних об'єктів і рішення завдань геометричного моделювання.

1. ОСНОВНІ ВІДОМОСТІ ПРО КОМП'ЮТЕРНУ ТЕХНІКУ І СПОСОБИ ЇЇ ВИКОРИСТАННЯ

1.1. Загальні положення

Слово комп'ютер означає обчислювач, тобто пристрій для обчислень. Це пов'язано з тим, що перші комп'ютери створювалися саме з цією метою, грубо кажучи, спочатку це були вдосконалені, автоматичні арифмометри. Принципова відмінність комп'ютерів від арифмометрів та інших рахункових пристроїв полягає в тому, що перші можуть виконувати тільки окремі обчислювальні операції (додавання, віднімання, множення, ділення і т. д.), а комп'ютери здатні без участі людини проводити складні послідовності обчислювальних операцій за наперед заданою інструкцією – програмою. Крім того, для зберігання даних, проміжних і підсумкових результатів обчислень, а також самих програм комп'ютери мають дуже важливу функцію – пам'ять.

Хоча комп'ютери створювалися для числових розрахунків, але виявилось, що вони також можуть обробляти інші види інформації, яку можна подати в числовій формі. Для обробки різної інформації на комп'ютері передбачено засоби, які перетворюють будь-який вид інформації в числову форму й навпаки. У наш час комп'ютери виконують не тільки обчислювальну функцію, їх використовують у видавничій практиці, вони здатні створювати різноманітні рухомі й статичні зображення, музичні твори, здійснювати управління підприємствами і т.д. Комп'ютери перетворилися на універсальні засоби для обробки всіх видів інформації, яка потрібна людині.

1.2. Історія розвитку комп'ютерів

Ще в першій половині XIX століття англійський математик Чарльз Беббідж спробував побудувати універсальний обчислювальний пристрій, тобто комп'ютер. Саме Беббідж уперше встановив, що комп'ютер повинний мати пам'ять, а для керування його функціями потрібна програма. Беббідж хотів побудувати свій комп'ютер як механічний пристрій, а програми збирався задавати за допомогою перфокарт - аркушів з цупкого паперу, які містять інформацію у вигляді отворів. На рис. 1.1 представлений проект обчислювальної машини Бебиджа. Проте довести до кінця цю роботові Беббідж не зміг - вона виявилася дуже доладною для того часу. Слід зазначити і першу в історії людства програмістку Аду Лавлейс, дочку великого англійського поета Джорджа Гордона Байрона, яка тісно співробітничала з Беббіджом.

У 40-х роках XX століття деяки дослідники впровадили ідеї Беббіджа за допомогою електромагнітних реле. Деремо з них був німецький інженер Конрад Цузе, який у 1941 р. побудував невеликий комп'ютер на основі безліч електромагнітних реле. Але події другої світової війни завадили цьому вченому продовжити свої дослідження. У 1943 р. на одному з американських підприємств її працівник Говард Ейкен створив потужний на той час

обчислювальний прилад під назвою "Марк-1". Він вже дозволяв проводити обчислення в сотні разів швидше, ніж за допомогою арифмометра. Проте недоліком цього комп'ютера була низька його надійність.

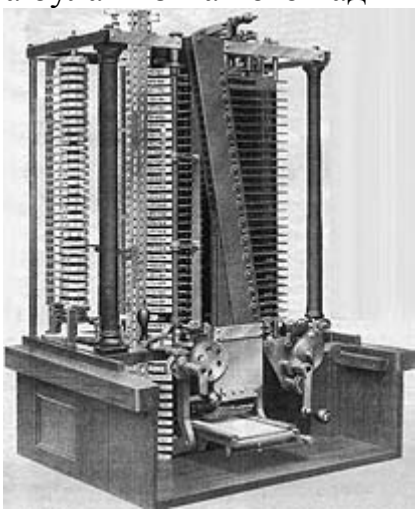


Рис. 1.1 Проект обчислювальної машини Чарльза Беббіджа

У 1945 р. американські інженери Джон Мочлі і Преспер Екерт сконструювали комп'ютер ENIAC (рис. 1.2) на основі електронних ламп. Створений ними прилад працював у тисячу разів швидше, ніж Марк-1, проте він більше простоював, ніж процював, через великі витрати часу на з' єднання провідників для введення початкових даних і виконання програми.

Рис. 1.2 ЕОМ " ENIAC "

У 1945 р. до роботи над створенням досконалої конструкції комп'ютера було залучено відомого математика Джона фон Неймана, який ясно і просто сформулював загальні принципи функціонування таких засобів. Перший комп'ютер, у якому були втілені принципи фон Неймана, був побудований у 1949 р. англійським дослідником Морісом Уїлксом.

У 40-х і 50-х роках минулого століття комп'ютери створювалися на основі електронних ламп, вони були дуже великими, дорогими і ненадійними.

У 1948 р. були винайдені транзистори, що замінили електронні лампи. Перші комп'ютери на основі транзисторів з'явилися в кінці 50-х років. Випущений у 1965 р. американською фірмою Digital Equipment міні-комп'ютер

PDP-8 за розміром нагодував холодильник і коштував всього 20 тис. дол. (тоді як комп'ютери 40 – 50-х років коштували мільйони доларів).

У 1959 р. Роберт Нойс (майбутній засновник фірми Intel) винайшов спосіб створення інтегральних схем, що вміщують у собі велику кількість транзисторів та інших напівпровідникових елементів. У 1968 р. фірма Burroughs випустила перший комп'ютер на інтегральних схемах, а в 1970 р. фірма Intel почала продавати інтегральні схеми пам'яті.

Того самого року один з працівників фірми Intel Маршіан Едвард Хофф сконструював інтегральну мікросхему, аналогічну за своїми функціями центральному процесору великого комп'ютера. Це був перший мікропроцесор Intel - 4004, продаж якого було разпочато в 1971 р. Через деякий час, у 1973 р. фірма Intel випустила 8-розрядний мікропроцесор Intel - 8008, а в 1974 р. його вдосконалену версію Intel - 8080, яка до кінця 70-х років стала стандартом для мікрокомп'ютерної індустрії.

Випуск першого персонального комп'ютера на основі мікропроцесора Intel - 8080 датується 1975 роком. Ціна цього приладу становила 500 дол. Його можливості булі вельми обмежені зважаючи на відсутність клавіатури і дисплея.

А вже в кінці 1975 р. програмісти Пол Аллен і Білл Гейтс (майбутній засновник фірми Microsoft) створили інтерпретатор мови Basic, що дозволило користувачам досить просто спілкуватися з комп'ютером і писати для нього програми.

У 1981 р. випущений новий персональний комп'ютер IBM PC з 16-розрядним мікропроцесором Intel-8088. Його використання дозволило значно збільшити потенційні можливості комп'ютера, оскільки він був разрахований на 1 мегабайт пам'яті, а комп'ютери, які існували до цих пір, були обмежені 64 кілобайтами.

Нині випускаються комп'ютери з характеристиками, що значно перевищують характеристики першого персонального комп'ютера IBM PC. Наприклад, оперативна пам'ять доведена до декількох гбайт, пам'ять жорстких дисків складає сотні і тисячі гбайт, а число розрядів комп'ютерів дорівнює числу 64 і більш.

У перспективі передбачається розвиток штучного інтелекту на основі оптико-лазерных технологій і застосування надвеликих інтегральних схем. Планується створити комп'ютер з великою швидкодією, величезним по потужності процесором і необмеженою віртуальною пам'яттю.

Як основний елемент для електричних ланцюгів буде використаний арсенід галия. Робота цих комп'ютерів буде заснована на паралельних обчисленнях.

Основоположниками советской вычислительной техники были Сергей Лебедев и Исаак Брук.

В 1951 р. в Києві під керівництвом академіка С. О. Лебедева була створена перша радянська ЕОМ, що дістала назву “МЭСМ” - мала електронна рахункова машина, яка мала 600 електронних ламп. Це був перший у континентальній Європі комп'ютер.

У 1952 р. в Москві під керівництвом Лебедева побудована швидкодіюча електронна рахункова машина “БЭСМ- 1” - на той час найпродуктивніша машина в Європі і одна з кращих у світі. Паралельно з ними створювалися електронно- обчислювальні машини Стріла, Урал, Мінськ, Раздан, Наири. На рис. 1.3 представлена ЕОМ "Урал-1", а на рис. 1.4 ЕОМ "Мир".

Рис. 1.3 ЕОМ "Урал-1"

Завдяки діяльності радянських учених С. О. Лебедева, В.М. Глушкова, М. В. Келдыша, М. О. Лаврентьєва, І.С. Брука, М. О. Карцева, Б.І. Рамєєва, В. С. Антонова, А.Н. Невського, Б.І. Буркова і керованих ними колективів Радянський Союз вирвався в число лідерів обчислювальної техніки, що дозволило в короткі терміни вирішити важливі науково-технічні завдання опанування ядерної енергії і дослідження космосу.

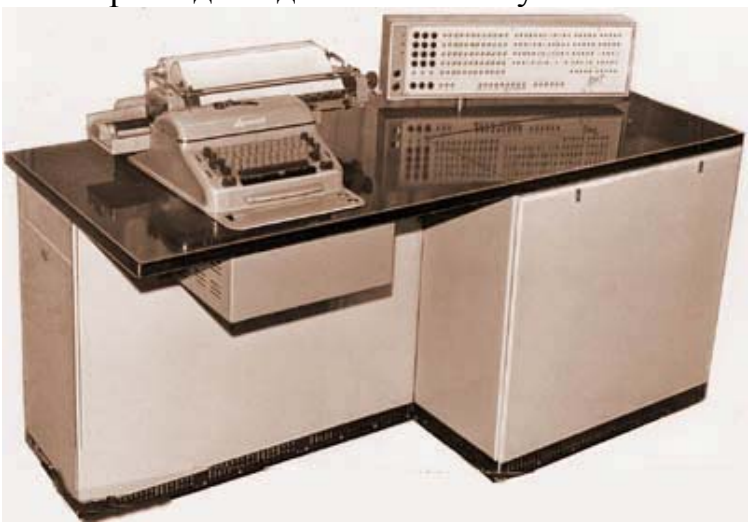


Рис. 1.4 ЕОМ "Мир"

З персональних комп'ютерів відомими в СРСР є комп'ютери серії

"Електроніка " (рис. 1.5), "ДВК" (рис. 1.6), "ЄС", "Іскра".

Рис. 1.5 Комп'ютер "Электроника ВК-0011М"

Рис. 1.6 Класичний варіант комп'ютера "ДВК-2"

1.2.1. Покоління ЕОМ

Уся історія розвитку людського суспільства пов'язана з накопиченням та обміном інформацією (наскальний живопис, писемність, бібліотеки, пошта, телефон, радіо, рахівниці і механічні арифмометри та ін.). Корінний перелом в технології обробки інформації почався після Другої світової війни. В обчислювальних машинах першого покоління основними елементами були електронні лампи. Ці машини займали величезні зали, важили сотні тонн і витрачали сотні кіловат електроенергії. Їх швидкодія та надійність були низькими, а вартість досягала 500 – 700 тисяч доларів.

Поява потужних і дешевших ЕОМ другого покоління стала можливою завдяки винаходу в 1948 році транзисторів. Головний недолік машин першого і другого поколінь полягав у тому, що вони складались із з великого числа компонентів, що сполучаються між собою. Точки з'єднання (паяння) були дуже ненадійними місцями в електронній техніці, тому ці ЕОМ часто виходили з ладу.

В ЕОМ третього покоління (із середини 60-х років XX століття) почали використовуватися інтегральні мікросхеми (чипи) – пристрої, що містять у собі тисячі транзисторів та інших елементів, але виготовляються як єдине ціле, без зварних, або паяних з'єднань цих елементів між собою. Це привело не тільки до різкого збільшення надійності ЕОМ, але й до сниження розмірів, енергоспоживання й вартості (до 50 тисяч доларів).

Історія ЕОМ четвертого покоління почалася в 1970 році, коли нікому не відома раніше американська фірма INTEL створила велику інтегральну схему (БІС), що містить у собі практично всю основну електроніку комп'ютера. Ціна однієї такої схеми (мікропроцесора) становила всього кілька десятків доларів, що й зумовило зниження цін на ЕОМ до рівня доступних для широкого кола користувачів.

П'яте покоління (1986 р. і до теперішнього часу) значною мірою визначається результатами роботи японського Комітету наукових досліджень в області ЕОМ, опублікованими у 1981р. Згідно з цим проектом ЕОМ і

обчислювальні системи п'ятого покоління окрім високої продуктивності і надійності при нижчій вартості за допомогою новітніх технологій, повинні задовольняти наступним якісно новим функціональним вимогам:

- забезпечити простоту застосування ЕОМ шляхом реалізації систем введення/виведення інформації голосом, а також діалогової обробки інформації з використанням природних мов;
- забезпечити можливість навчаної, асоціативних побудов і логічних висновків;
- спростити процес створення програмних засобів шляхом автоматизації синтезу програм по специфікаціях початкових вимог на природних мовах;
- поліпшити основні характеристики і експлуатаційні якості обчислювальної техніки для задоволення різних соціальних завдань, поліпшити співвідношення витрат і результатів, швидкодії, легкості, компактності ЕОМ;
- забезпечити різноманітність обчислювальної техніки, високу адаптується до додатків і надійність в експлуатації.

1.3. Методи подання відображення електронної інформації

1.3.1. Числова форма

Комп'ютер може обробляти інформацію, тільки перетворену в *числову форму*. Уся інша інформація (звуки, зображення, покази приладів і т. д.) для обробки на комп'ютері теж має бути перетворена в числову форму. Наприклад, якщо це стосується звуку, то можна через невеликі проміжки часу вимірювати його інтенсивність, і результати цих вимірювань будуть являти собою числову форму подібного роду інформації.

1.3.2. Кодування символів

Для обробки текстової інформації зазвичай при введенні в комп'ютер кожна буква кодується певним числом, а при виведенні на зовнішні пристрої (екран або друк) для сприйняття людиною за цими числами будуються відповідні зображення букв. Відповідність між набором букв і числами називається *кодуванням символів*. Правила встановлення відповідності записують у таблицю, яка називається *кодовою*. Кодова таблиця являє собою запис, що встановлює відповідність між символами алфавіту та двійковими числами. Ці числа називаються *кодами символів* і відповідають внутрішньому зображенню символів у комп'ютері. Кодову таблицю називають також кодовою сторінкою. Яким чином працює кодова таблиця? Коли користувач натискає яку-небудь клавішу на клавіатурі, електронна схема клавіатури формує певний двійковий код. Наприклад, якщо натиснута клавіша цифри "**1**", то сформується двійковий код **00110001**. Натиснення на клавішу "**2"** відповідає коду **00110010**. Залежно від натиснутої клавіші формується той чи інший двійковий код, який задається кодовою таблицею. За основу кодування символів у персональних

комп'ютерах використовується кодова таблиця **ASCII**. ASCII – це скорочений варіант американського стандарту кодів для обміну інформації (American Standard Code for Information Interchange). У цій таблиці кожен символ кодується двійковим числом, яке складається із семи розрядів. Для кожного алфавіту розробляється своя кодова сторінка. Перша половина таблиці

ASCII – це стандартні коди і обов'язкові для всіх кодових сторінок. Наступні коди – з 128 до 255 (друга половина таблиці) віддаються в розпорядження для розробки стандартів окремих країн.

1.3.3. Системи числення

У цифрових обчислювальних машинах інформація задається у вигляді цифрових або буквених символів. При цьому використовується не будь-який набір символів, а певна система. У електронних обчислювальних машинах застосовуються позиційні системи числення. Як правило, усі числа в середині комп'ютера відображаються за допомогою нулів і одиниць, а не звичайних десяти цифр. Значення величини можна представляти як брехня або істинність якого-небудь твердження (0 - брехня, 1 - істина), відповідно найменша одиниця інформації називається - біт. Про те за допомогою одного біта неможливо представити цифри десяткової системи числення або букви алфавіту, тому для представлення символів використовується декілька біт. Величина, що містить 8 біт, називається байтом.Одного байта вже досить для представлення основних чисел і букв. У байт можна записати 256 різних комбінацій нулів і одиниць - це дозволяє закодувати 256 різних символів. Більшими одиницями виміру є:

1 Кбайт = 1024 Байт,

1 Мбайт = 1024 Кбайт,

1 Гбайт = 1024 Мбайт,

1 Тбайт = 1024 Гбайт.

Коди символів задаються тут за допомогою кодировочной таблиці - кодування (для кожного коду вказується відповідний символ).

Комп'ютери зазвичай працюють у двійковій системі числення, оскільки при цьому їхня будова виходить значно простішою. Введення чисел у комп'ютер і виведення їх для читання може здійснюватися в звичній для людей десятковій системі - усі необхідні перетворення можуть виконувати програми, що працюють на комп' ютері.

Ця система числення називається позиційною, тому що значення кожної цифри, яка входить у число, залежить від її положення в записі числа. Позиційні системи числення бувають різними залежно від основи: десяткові з основою десять, вісімкові з основою вісім, двійкові з основою два й так далі.

Розглянемо десяткову систему числення. Для її зображення використовують цифри: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**. Число десять є складеним. Кожне десяткове число можна розкласти на ступенях основи десяткової системи числення. Наприклад, число **5213,6** можна уявити записати як поліном, кожен член якого є добутком коефіцієнта на основу системи певною мірою:

5213,6 = 5•10³ + 2•10² + 1•10¹ + 3•10⁰ + 6•10⁻¹, де **5, 2, 1, 3, 6** є коефіцієнтами.

У двійковій системі числення коефіцієнтами виступають цифри **0** і **1**. Основою системи є число **2**. Число **2** зображається у вигляді **10**, оскільки **10** = **1•2¹ + 0•2⁰**

У загальному вигляді число у двійковій системі буде записано таким чином: $A_2 = r_n \cdot 2^n + r_{n-1} \cdot 2^{n-1} + \cdots + r_1 \cdot 2^1 + r_0 \cdot 2^0$, де r_i - коефіцієнт, який набуває значення одиниці або нуля.

Дуже часто при описі оброблюваних комп'ютером даних умісту оперативної пам'яті та в інших випадках використовується шістнадцяткова система числення. Вона зручна тим, що дуже просто співвідноситься із двійковою системою, у якій працює комп'ютер, при цьому шістнадцяткова цифра відповідає чотирьом двійковим розрядам. Для шістнадцяткових цифр використовуються такі позначення: **A** – десять, **B** – одинадцять, **C** – дванадцять, **D** – тринадцять, **E** – чотирнадцять, і **F** – п'ятнадцять. Для позначення того, що число записане в шістнадцятковій системі числення, в кінці додають символ "**h**" або "**H**" (**h** – перша буква слова hexadecimal, тобто шістнадцятковий). Наприклад, **B9h = 11•16 + 9 = 185**.

1.4. Основні принципи роботи комп'ютера

У основу побудови переважної більшості ЕОМ покладені наступні загальні принципи, сформульовані в 1945 році американським ученим угорського походження Джоном фон Нейманом:

1. Принцип двійкового кодування - згідно з цим принципом уся інформація, що поступає в ЕОМ, кодується за допомогою двійкових сигналів.
2. Принцип програмного управління - з нього виходить, що програма складається з набору команд, які виконуються процесором автоматично один за одним в певній послідовності.
3. Принцип однорідності пам'яті - програми і дані зберігаються в одній і тій же пам'яті. Тому ЕОМ не розрізняє, що зберігається в цьому елементі пам'яті - число, текст, або команда. Над командами можна виконувати такі ж дії, як і над даними.
4. Принцип адресності - структурно основна пам'ять складається з пронумерованих осередків.

Схема побудови ЕОМ, згідно з принципами фон Неймана представлена на рис. 1.7, а її елементи виконують такі функції:

- ***арифметично-логічний*** пристрій відповідає за арифметичні й логічні операції;
- ***пристрій керування*** організовує процес виконання програм;
- ***оперативна пам'ять*** (запом'ятовальний пристрій) зберігає програми і дані;
- ***зовнішні пристрої*** вводять або виводять інформацію.

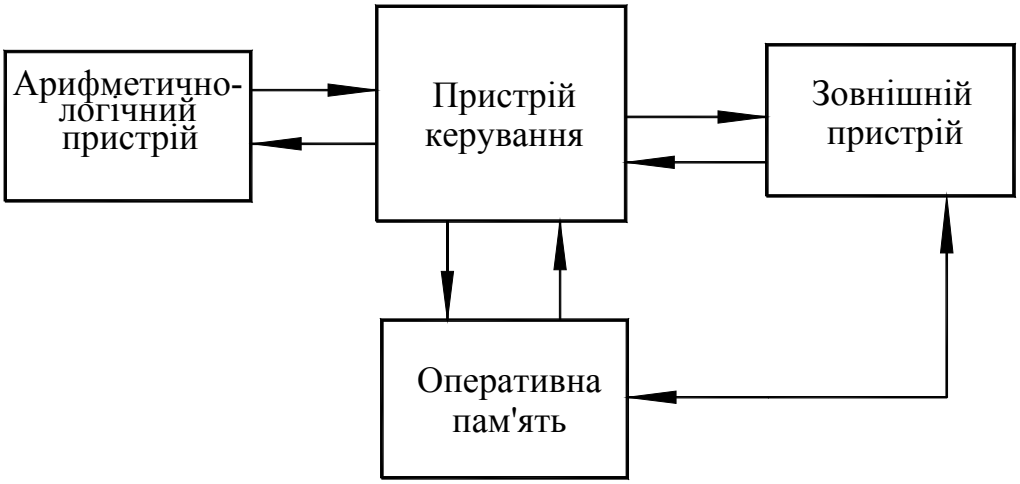


Рис. 1.7. Загальна схема комп'ютера, побудованого за принципом фон Неймана

Сучасна ЕОМ – це складна система, в яку входять такі електронні, електромеханічні та механічні пристрої:

- запам'ятовувальний (або просто пам'ять);
- процесор, що включає пристрій керування і арифметично-логічний;
- пристрій уведення даних;
- пристрій виведення даних.

Ці пристрої з'єднані магістралями зв'язку, по яких передається інформація.

Схему основних пристроїв ЕОМ і найважливіших зв' язків між ними подано на рис. 1.8.

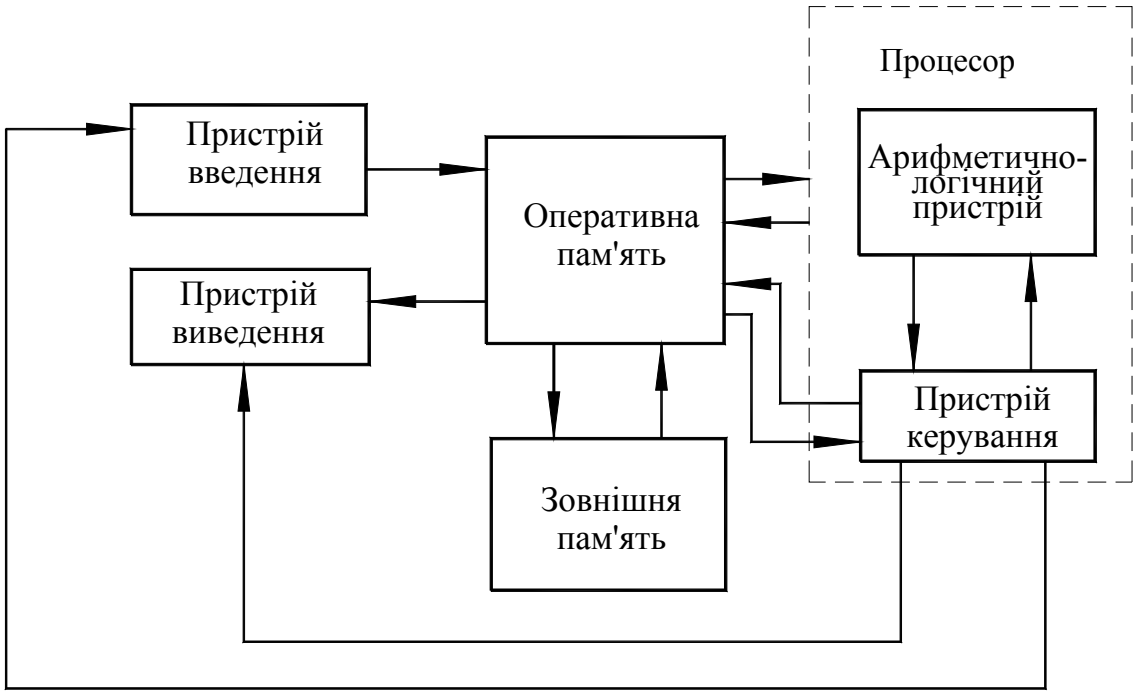


Рис. 1.8. Схема основних пристроїв ЕОМ та найважливіших зв'язків між ними

Запам'ятовувальний пристрій (**ЗУ**) служить для прийому інформації з інших пристроїв, її запам'ятовування, а також видачі необхідної інформації в інші пристрої машини.

Пам'ять будь-якої ЕОМ зазвичай складається з двох частин – оперативної та зовнішньої. В оперативну пам'ять можна швидко записати інформацію і швидко її звідти вилучити.

До оперативної пам'яті зазвичай пред'являють дві основні вимоги: швидкодія та великий обсяг. Певною мірою ці вимоги вступають у протиріччя, і тому важко створювати пам'ять машини, яка за обома цими параметрами задовольняла б користувачів ЕОМ. І хоча в сучасних ЕОМ обсяг оперативної пам'яті досить великий, проте її не завжди вистачає для розв'язку великих задач. Тому в усіх обчислювальних машинах, починаючи з машин першого покоління, використовують зовнішню пам'ять.

У зовнішню пам'ять зазвичай записують ту частину інформації, яка рідше використовується в ході розв'язування задач.

Якщо звернутися до аналогій, то оперативну пам'ять можна порівняти із запом'ятовувальною функцією мозку, а зовнішню пам'ять – із записником, довідником, книгою, тобто відокремленими від людини об'єктами, відомості з яких можна передати іншій особі.

Уся оперативна пам'ять складається з чарунок. Чарунка використовується для зберігання одного машинного слова, яким може бути число, команда або деяке допоміжне слово. Обсяг пам'яті визначається кількістю чарунок. Усі чарунки рівноправні і однаково доступні для інших пристроїв ЕОМ. Коли в яку-небудь чарунку записується слово, то воно передбуває там до тих пір, поки в цю чарунку не буде внесене нове слово. Тоді старе автоматично зникає, і ніяких накладок нового слова на старе не відбувається, просто старе стирається і пишеться нове.

Принцип дії зовнішніх запом'ятовувальних пристроїв полягає в тому, що магнітний носій інформації механічно переміщається під головками магнітного запису або прочитування. Тому, перш ніж дістатися до потрібної інформації, доводиться чекати її появи під головками. В очікуванні такого переміщення проходить досить багато часу, через що зовнішні пристрої відносять до типу засобів з так званим послідовним доступом, тобто вони не передбачають прямого звернення до будь-якої необхідної інформації, бо з цією метою треба послідовно виконати певні дії. У цьому полягає ще одна відмінність зовнішніх запом'ятовувальних пристроїв від оперативних.

Переробка інформації відбувається в ***арифметично-логічному пристрої*** ЕОМ. Там виконуються арифметичні і логічні операції. Окрім цих операцій, арифметично-логічний пристрій формує деякі керуючі сигнали, що дозволяють машині залежно від результатів обчислень вибирати шлях для подальших дій.

У кожній операції бере участь спеціальний запам'ятовувальний пристрій, який називається регістром суматора. Інформацію для виконання операцій, тобто операнди, арифметично-логічний пристрій отримує з оперативної пам'яті ЕОМ. Результат операції може залишитися в арифметико-логічному пристрої, або пересилатися на зберігання.

Один з найважливіших в ЕОМ – ***пристрій керування***. З самої назви зразуміле його призначення – керування процесом роботи ЕОМ. Він працює за заданою програмою, вибираючи інформацію із запом'ятовувального та арифметично-логічного пристроїв.

Пристрій введення служить для внесення в пам'ять ЕОМ необхідної інформації – програми і початкових даних. Введення інформації та керування процесом виконання програм здійснюється з перфокарт і терміналів. Роль терміналів виконують засоби, до складу яких входить телевізійний екран і клавіатура. На екрані можна перевірити правильність введення інформації, простежити за процесом розв'язування задачі, в потрібний момент внести необхідні корективи, подивитися на результати виконання програми.

Пристрої виведення відображають результати виконання програми на перфокарті, перфострічці, дисплеї, принтері.

1.5. Різновиди програм для комп'ютерів

Комп'ютер – це універсальний прилад для переробки інформації. Але сам по собі комп'ютер – це просто ящик з набором електронних схем. Він не володіє знаннями в жодній сфері свого застосування. Усі ці знання зосереджуються у виконуваній на комп'ютері програмі. Для того, щоб комп'ютер міг виконати певні дії, необхідно скласти програму, тобто точну і докладну послідовність інструкцій, яким чином треба обробляти інформацію. Такі інструкції мають бути написані зразумілою для комп'ютера мовою.

Програми, які використовуються в комп'ютерах можна поділити на три види:

● ***прикладні,*** що безпосередньо забезпечують виконання необхідних користувачам робіт: обробку інформації, редагування тексту, виконання креслень та інших зображень;

● ***системні***, які виконують різні допоміжні функції, наприклад перевірку працездатності пристроїв комп'ютера, створення копій використовуваної інформації і т. д. Особливу роль серед системних програм відіграє операційна система – програма, що керує комп'ютером і запускає всі інші програми;

● ***інструментальні системи*** (системи програмування), що забезпечують створення нових програм для комп'ютера.

1.6. Характеристика основних пристроїв комп'ютера

Кожний персональний комп'ютер складається з трьох основних частин:

● системного блока;

● клавіатури, що дозволяє вводити в комп'ютер символи;

● монітора, який показує текстову і графічну інформацію.

Охарактеризуємо кожну з цих частин.

Системний блок, містить такі основні вузли:

● електронні схеми, керівники роботою комп'ютера (мікропроцесор, оперативна пам'ять, контролери пристроїв і т. д.);

● блок живлення, що перетворює змінний струм електромережі в постійний струм низької напруги;

● накопичувачі даних на жорсткому магнітному диску;

● інші пристрої.

Більшість (90%) сучасних комп'ютерів являються IBM-сумісні персональні засоби, це значить, що їхня робота узгоджується комп'ютером IBM PC, який створений у 1981 р. найбільшою у світі комп'ютерною фірмою IBM.

Системний блок складається з таких елементів:

● ***КЕШ – пам'ять*** служить для прискорення доступу до інформації, що містить оперативна пам'яті (ОП). Цей елемент перебуває "між" мікропроцесором і ОП. Він містить копії тих даних, які найбільш часто використовуються в ОП. При зверненні мікропроцесора до пам'яті спочатку пошук потрібних даних ведеться в кеш-пам'яті, оскільки для цього потрібно набагато менше часу;

● ***BIOS -*** пристрій з наявністю постійної пам'яті, у якові занесене первинні незмінні дані. Програми, що виконуються, можуть їх тільки читати. Такий вид пам' яті має назви - **ROM** (read only memory) або **ПЗП** (постійний запам'ятовуючий пристрій). У BIOS включене програми для перевірки обладнання комп'ютера та програма завантаження операційної системи (ОС) а також програми з обслуговування пристроїв комп'ютеру;

● ***Відеопам'ять*** – це засіб для зберігання зображення, які виводяться на екран монітора. Ця пам'ять зазвичай входить до складу відеоконтролера – електронної схеми, яка керує виведенням зображення на екран монітора.

Кожна електронна плата в системному блоці являє собою плоский шмат пластику, на якому напаяні діоди, транзистори, резистори, конденсатори і різні рознімні з'єднання. У середині електронної плати прокладено провідники для з'єднання компонентів плати між собою.

Найбільшою електронною платою в комп'ютері є ***системна*** або ***материнська плата***. На ній звичайно розташовані мікропроцесор, оперативна пам'ять, кеш-пам'ять, шина і BIOS. Крім того на цій платі розміщено електронні схеми (контролери), які керують деякими пристроями комп'ютера. Так контролер клавіатури завжди розміщується на материнській платі. До того ж можуть бути і контролери жорстких дисків, дисководів. Але більшість контролерів розташовані на окремих електронних платах. Ці плати вставляються в спеціальні рознімні з'єднання (слоти) на материнській платі.

За допомогою доповнення і заміни плат контролерів користувач може модифікувати комп'ютер. Наприклад, можна доповнити комп'ютер факс-модемом, звуковою картою, платою прийому телепередач і т. д.

● ***Шини***. Вставляючи контролер у рознімні з'єднання материнской плати, його підключають до шини-магістралі передачі даних від ОП до інших контролерів. Сучасні комп'ютери звичайно обладнені двома шинами, а саме:

• шина ***ISA*** призначена для контролерів низькошвидкісних пристроїв (тобто для обміну даними між клавіатурою, мишею , дисководами для дискет, модемом, звуковою картою і т. д.);

• шина ***PCI*** – для обміну даними з високошвидкісними пристроями (жорсткими дисками, відеоконтролером).

Роботу шин забеспечують такі пристрої:

• блок живлення, який перетворює змінний струм мережі у постійний струм низької напруги;

• накопичувачі (або дисководи) для гнучких магнітних дисків;

• накопичувач на гнучкому магнітному диску;

• інші пристрої.

1.7. Пристрої, що підключаються до комп'ютера, їх класифікація та застосування

● ***Вказівні пристрої.*** Дозволяють показувати або позначати ті чи інші елементи на екрані комп'ютера.

● ***Маніпулятор- миша.*** Пристрій з двома або трьома кнопками. При переміщені миші по площині на екрані монітора відповідним чином переміщується вказівка миші. Для виконання тієї чи іншої дії потрібно натиснути відповідну кнопку.

● ***Дисковод для дискет.*** Пристрій для запису і зчитування інформації, що існує на компютере або дискеті. Дискети мають ємність 1.44 Мб.

● ***Монітор і відеокамера.*** Монітор (дисплей) комп'ютера використовується для виведення на екран текстової і графічної інформації. Різні монітори мають неоднакову кількість точок на зображенні від 640×480 до 1600×1280. Електронні схеми , які забезпечують формування відеосигналу і тим самим визначають зображення, називається ***відеконтролером***. Відеоконтролер має вигляді спеціальної плати, яка вставляється у рознімне з'єднання системної шини, але інколи він не входить до складу материнської плати. Відеоконтролер може працювати у текстовому або графічному режимі. У текстовому режимі монітор умовно розбивається на знакомісця з 25 рядків по 80 символів кожен. У кожне знакомісце може вводитись один із 256 символів. Для роботи в графічному режимі зображення виводиться у вигляді прямокутної сітки точок, колір кожної з яких задається окремо.

● ***Жорсткий диск (***вінчестер***).*** Пристрої призначені для постійного зберігання інформації, потрібної при роботі з комп'ютером: програм ОС, пакетів програм, що постійно використовуються, редакторів документів і т. д.

● ***Дисковод*** для компакт- дисків. Це пристрий для прийому і считування інформації, CD - ROM та DVD - ROM). Компакт-диски містять десятки Гб інформації.

● ***USB -флеш-накопичувач*** - пристрій, що запам' ятовує та використовує як носія флеш-пам'ять;

● ***Принтери*** (друкуючий пристрій). Призначений для виведення інформації на папір (тверда копія).

Види принтерів*:*

• ***Матричні***

Принцип дії: друкуюча головка принтера містить вертикальний ряд тонких металевих стержнів (голок). Головка рухається вздовж рядка, який друкується,

а стержні в потрібний час ударяють по аркушу через плівку, утворюючи зображення символів.

• ***Струменеві***

У принтерах цього виду зображення формується мікрокаплями спеціального чорнила, яке викидається на аркуш через сопла в друкуючій голівці. Як і в матричних принтерах, друкуюча голівка струменевого принтера рухається по горизонталі, а наприкінці руху кожної горизонтальної смужки зображення папір просувається по вертикалі. На відміну від матричних принтерів, робота струмневих супровджується меншим шумои, а якість друку набогато вище.

• ***Лазерні***

Забезпечують найкращу (близьку до поліграфічної) якість чорно-білого та кольорового друку. В лазерних принтерах використовується принцип ксерографії: зображення переноситься на папір зі спеціального барабана, до якого електрично притягаються часточки фарби (тонера). Відмінність від звичайного копіювального апарата полягає в тому, що друкуючий барабан цього принтера електризується за допомогою лазера, виконуючи команди комп'ютера.

● ***Модеми і факс-модеми.*** Ці пристрої дають можливість працювати в глобальних електронних мережах (Internet), використовуються також для роботи с електронною поштою, дозволяють за допомогою комп'ютера надсилати і приймати факси і т. д.

● ***Засоби мультимедіа.*** "**Мультимедіа**" означає здатність комп'ютера опроцьовувати інформацію різних видів, а не тільки цифровою. Насамперед це стасується звукової та відеоінформації. Іншими словами, мультимедійні засоби можуть відтворювати:

• музику, мову, різноманітну звукову інформацію;

• фільми та іншу відеоінформацію.

Щоб працювати з мультимедіа програмами комп'ютер повинен бути забезпечений дисководом для компакт-дисків, звуковою картою та акустичною системою (колонки, навушники, мікрофон).

● ***Клавіатура.*** **–** це пристрій, що являє собою сукупність разміщених у певному порядку клавішів на панелі. Клавіатура дає змогу виконувати на комп'ютері різноманітні операціїї. Розглянемо особливості цього пристрою. Клавіші керування курсором розташовуються у правій частині клавіатури, вони, як правило, продубльовані. Клавіші із стрілками переміщають курсор на одну позицію у відповідному напрямку.

Клавіші **Home** і **End** переміщають курсор в початок і кінець рядка відповідно.

Клавіші **PgUp** і **PgDn** перегортають текст на екрані на сторінку вгору і вниз відповідно.

Клавіша **Del** (**Delete**) використовується для видалення символа, що стоїть під курсором. Клавіша **Ins** (**Insert**) призначена для перемикання між двома режимами введення символів: вставленням і заміною.

Клавіша **Enter –** одна з найголовніших у комп'ютері. Вона призначена для закінчення введення рядка. Натисненням цієї клавіші закінчується введення кожної команди операційної системи DOS.

Клавіша **Esc** (**Escape**), як правило, використовується для відміни якої-небудь дії, виходу з режиму програмування і т. д.

Клавіша **Tab** (**табуляція**) використовується для переходу до наступної позиції табуляції. Іноді вона використовується як перемикач між вікнами на екрані і т. д.

Клавіша **Backspace** (стрілка вліво над клавішею **Enter**) видаляє символ, що стоїть зліва від курсора.

Клавіша **Shift** призначена для введення великих букв алфавіту й інших символів, розміщених на верхньому регістрі клавіатури. Щоб ввести такий символ або велику букву, потрібно натиснути цю клавішу і, не відпускаючи її, натискувати потрібну клавішу.

Клавіша **Caps Lock** фіксує введення великих букв алфавіту. Це зручно, коли виникає потреба в їхній великій кількості. Повторне натиснення цієї клавіші відміняє режим введення великих букв.

Клавіші **Alt** і **Ctrl** призначені для зміни значення інших клавіш.

Клавіша **Num Lock** (**блокування цифр**) вмикає і вимикає режим, при якому натисненням на клавіші керування курсором вводяться цифри 1 – 9, 0, і крапка. Цей режим зручний для введення чисел.

Клавіша **Pause** припиняє роботу програми.

1.8. Особливості експлуатації комп‘ютера

● ***Вмикання комп‘ютера.*** Якщо комп‘ютер підключений до мережі через пристрій безперебійного живлення (UPS), то спочатку треба ввімкнути цей пристрій. Якщо комп'ютер підключений через стабілізатор напруги, то вмикаємо стабілізатор.

Потім вмикаємо монітор, зовнішні пристрої, які нам потрібні (принтер, модем і т. д.), потім вмикаємо сам комп'ютер, натискуючи кнопку або клавішу на його корпусі (**Power**).

Потім на екрані комп‘ютера з‘являються повідомлення про хід роботи програм, які виконують перевірку його обладнання і початкове завантаження операційної системи комп‘ютера. При цьому підчас початкового завантаження вміст ОП очищується, після чого автоматично стартують згадані програми, які входять у постійну пам'ять **BIOS.** Якщо ці програми знаходять помилку, то виводять її код помилки на екран монітора. Причому некритична помилка дає можливість користувачеві продовжити роботу після натисненні клавіші **F1**. Якщо ж помилку визнано критичною, то процес завантаження зупиняється.

У разі, коли помилку виявлено ще до перевірки відеосистеми комп‘ютера, то про її повідомляється не написом на екрані, а комбінацією звукових сигналів. Перелік звукових сигналів можна дізнатись із технічної документації, що супроводжує материнську плату комп‘ютера.

● ***Вимикання*** **комп‘ютера.** Для повного вимикання комп'ютера потрібно:

• закінчити роботу з усіма програмами;
• якщо ОС передбачає процедуру виходу, то виконати цю процедуру;
• вимкнути принтер та інші зовнішні пристрої, приєднані до комп‘ютера;
• вимкнути монітор.

● ***Перевантаження комп‘ютера.*** Ця операція виконується в разі “зависання”, тобто якщо комп‘ютер не реагує ні на жодні команди користувача.

Перевантаження виконується за допомогою комбінації клавіш ***Ctrl – Alt – Del,*** або натискуванням кнопки ***reset*** на корпусі комп‘ютера.

Контрольні питання

1. Яке значення має слово комп'ютер?
2. Як за принципом дії арифмометр відрізняється від комп'ютера?
3. У якому сторіччі було створено перший комп'ютер?
4. Як змінювалась елементна база комп'ютерів за всю історію їхнього розвитку?
5. Чим відрізняються комп'ютери різних поколінь?
6. Що таке кодова система і як вона використовується в комп'ютерах?
7. Які системи числення використовуються в комп'ютерах?
8. Як виглядає функціональна схема комп'ютера, побудованого за принципом Джона фон Неймана?
9. З яких основних пристроїв складається комп'ютер?
10. У чому полягає відмінність оперативної пам'яті від зовнішньої?
11. Які пристрої в комп'ютері призначено для введення інформації?
12. Які пристрої в комп'ютері призначено для виведення інформації?
13. Яку функцію виконує процесор?
14. Що таке материнська плата і її функції?
15. Охарактеризуйте системний блок комп'ютера, яке його призначення?
16. Що являє собою **BIOS**-память, з якою метою її використовують?
17. Що являє собою **Кеш**-пам'ять, особливості та призначення?
18. Охарактеризуйте призначення маніпулятора миші, які його різновиди?
19. Які накопичувачі інформації використовують у комп'ютері?
20. Які види принтерів використовують у роботі з комп'ютером, їхні переваги й недоліки?
21. Яке призначення на клавіатурі комп'ютера клавіш **Home, End, PgUp, PgDn, Del, Ins, Enter, Esc, Tab, Backspase, Shift, Caps Lock, Num Lock, Alt, Ctrl, Pause** ?

2. ОПЕРАЦІЙНІ СИСТЕМИ КОМП'ЮТЕРА

2.1. Призначення операційної системи

Операційна система (ОС) - це сукупність програмних засобів, що здійснюють керування ресурсами комп'ютера, запуск прикладних програм, їхню взаємодію із зовнішніми прибудовами й іншими програмами,

забезпечують діалог користувача з комп'ютером, а також керування файлами. З одного боку, ОС має у своїй основі базове програмне забезпечення комп'ютера, що входити у його систему BIOS (базова система уведення-виведення інформації), з іншого боку, вона сама є опорою для програмного забезпечення вищих рівнів - прикладних і більшості службових програм.

Основні функції ОС полягають у забезпеченні функціонування кількох видів інтерфейсу:

- користувальницького (зв'язок між користувачем та апаратними засобами комп'ютера);
- апаратно-програмного (між програмним і апаратним забезпеченням);
- програмного (між різними видами програмного забезпечення).

2.1.1. Забезпечення користувальницького інтерфейсу

Операційні системи здатні забезпечувати як пакетний, так і діалоговий режим роботи користувача. У пакетному режимі система автоматично виконує задану послідовність команд. Здатність ОС перервати поточну роботу й реагувати на дії, виконані користувачем за допомогою керуючих пристроїв сприймається як діалоговий режим роботи.

Сучасні дискові системи підтримують файлову систему, яка створює умови для зберігання даних на дисках, а також забезпечує доступ до цих даних.

2.1.2. Реалізація апаратно-програмного інтерфейсу

Гнучкість апаратних і програмних конфігурацій обчислювальних систем підтримується за рахунок того, що кожен розробник устаткування прикладає до нього спеціальні програмні засоби керування – драйвери. Надання основних засобів для обслуговування комп'ютера – одна з функцій ОС. Зазвичай вона реалізується за допомогою включення в базовий склад ОС першочергових службових додатків, серед яких можна назвати засоби перевірки “стискування” дисків; керування віртуальною пам'яттю; кешування дисків; резервне копіювання даних.

Сучасні ОС можуть включати мінімальні набори прикладного програмного забезпечення, які можна використовувати для виконання таких простих практичних завдань, як робота з текстовими документами, простими зображеннями та ін.

Із розвитком апаратних засобів обчислювальних систем і засобів зв'язку функції ОС безперервно розширюються, а способи виконання таких функцій удосконалюються.

Система Windows активно спирається на технологію ***Plug and Play*** і працює в основному з пристроями, які відповідають її принципам.

Plug and Play - це стандарт комп'ютерної індустрії для автоматизації процесу додавання нових можливостей до вашого комп'ютера. Технологія ***Plug and Play*** виникла у зв'язку з історичними проблемами, пов'язаними з

установками звукових карт на комп'ютери, що працювали під керуванням DOS або Windows 3.1.

Усі компоненти ***Plug and Play*** призначені для досягнення єдиної мети - забезпечити спільну автоматичну роботу комп'ютера, периферійних пристроїв і їх драйверів, а також операційної системи при мінімальному втручанні з боку користувача. Працюючи з системою, яка повністю відповідна вимогам ***Plug and Play***, користувачі можуть не турбуватися про те, чи виникне при установці нового пристрою апаратний конфлікт з іншим пристроєм.

2.1.3. Забезпечення програмного інтерфейсу

Робота із прикладними програмами являє собою найбільш важливу частину функціонування ОС. В аспекті керування виконанням додатків розрізняють однозадачні й багатозадачні системи. Однозадачні ОС (наприклад, MS DOS) передають усі ресурси обчислювальної системи одному виконуваному додатку і не допускають паралельного виконання іншого додатка. Проте більшість сучасних графічних ОС – багатозадачні. Вони забезпечують можливість одночасної або почергової роботи кількох додатків; можливість обміну даними між додатками, можливість сумісного використання програмних, апаратних, мережевих та інших ресурсів обчислювальної системи кількома додатками.

2.2. Операційна система MS-DOS

Операційна система MS-DOS була створена в 1981 р. фірмою Microsoft на замовлення фірми IBM, яка тоді розробляла комп’ютери IBM PC. Поява нових і більш потужних ОС не ліквідувала потребу в програмах MS-DOS і DOS. Це пояснюється неможливістю або економічною недоцільністю експлуатації більш потужних ОС. Тим більше, що при виході з ладу системи Windows (різних модифікацій) для діагностики й ліквідації збоїв використовують саме DOS.

Система MS-DOS складається із наступних частин:

Дискові файли ***IO.SYS*** і ***MSDOS.SYS*** (основні системні файли) містять програми MS-DOS, які постійно перебувають в оперативній пам’яті комп’ютера.

Ці файли повинні бути в кореневому каталозі диску, з якого відбувається завантаження системи.

Командний процесор ***DOS*** обробляє команди користувача. Стандартний командний процесор має назву ***COMMAND.COM***. Деякі команди користувача , такі як ***Type, Dir, Copy*** командний процесор виконує самостійно (вони називаються внутрішніми), для виконання інших (зовнішніх) процесор шукає на дисках програму, що відповідає цій команді, завантажує її в пам’ять і керування передає саме їй. Після закінчення роботи програми процесор вилучає її з пам’яті і видає запрошення.

Зовнішні команди ***DOS*** – це програми, які постачаються в комплекті з ОС у вигляді окремих файлів (форматування дискет, перевірка дисків). Ці програми записуються в окремий каталог.

Завантажник ***DOS*** – цю програму розміщено в першому секторі кожної дискети і в першому секторі логічного диску, з якого завантажується ***DOS***. Призначення цієї програми – завантаження даних у пам'ять системного файлу ***IO.SYS***.

2.2.1. Початкове завантаження операційної системи MS-DOS

Програма-завантажник операційної системи перевіряє наявність у кореневому каталозі завантажувального диска або дискети файлів ***IO.SYS*** і ***MSDOS.SYS***. Якщо ці файли не знайдено, виводиться повідомлення:

non-system disk or disk error. Replase and strike any key when ready.

Якщо ці файли наявні, то відбувається завантаження у пам'ять початку файлу ***IO.SYS*** і передає йому керування. У пам'яті розміщено програма, яка зчитує кінець файлу ***IO.SYS*** і файл ***MSDOS.SYS***. Потім з кореневого каталогу зчитується файл конфігурації ***CONFIG.SYS*** і виконуються всі дії, що в ньому записані. Якщо ж файл ***CONFIG.SYS*** відсутній, то всі параметри встановлюються за умовчуванням. Потім з кореневого файлу інформація вводиться в командний процесор і йому передається керування. Командний процесор здійснює командний файл ***AUTOEXEC.BAT***. Зміст повідомлень при завантаженні ***ОС***, що виводяться на екран монітора, залежить від версії ***DOS*** і змісту команд у файлах ***CONFIG.SYS*** та ***AUTOEXEC.BAT***.

Файл ***CONFIG.SYS*** містить спеціальні команди, які задають параметри ***DOS,*** а також перелік драйверів (програми, що розширюють можливості ОС), які необхідно завантажити в ОЗК.

Найбільш часто використовуються такі команди файла ***CONFIG.SYS***:

device= ім'я файлу драйвера;

dos=high – перемістити частину коду MS DOS у перші 64 Кілобайта розширеної пам'яті;

buffers=число буферів (для операцій введення-виведення);

files=число файлів – максимальне число одночасно відкритих файлів.

Наприклад:

device=c:\msdos\himem.sys –завантаження драйвера

buffers=40

files=50

Файл ***AUTOEXEC.BAT***, як правило, містить команди:

- запуску резидентних програм;
- встановлення змінних оточення ***DOS*** (команда ***SET***);
- ***Path*** – для задання списків каталогів, у яких відбувається пошук;
- ***Prompt*** – для встановлення формату запрошення ***DOS***.

Наприклад:

@ echo off

Path c:\exe;c:\exe\msdos;d:\word

prompt pg
rem запрошення містить назву поточного каталогу і символ “>”;
set temp=c:\windows\temp (для тимчасових файлів);
rem запуск драйвера клавіатури;
keyb ru , , c:\exe\msdos\keybrd2.sys
call c:\antivirus\virtest.bat
nc

2.2.2. Файлова система MS DOS. Поняття про каталог. Атрибути файлу

Файл – це пойменована ділянка пам’яті на диску або дискеті. У файлах можуть зберігатися тексти програм, документи, готові до виконання програми і т. д.

В іменах файлів і та їх типах можна використовувати ті сами символи, що і в іменах каталогів. Розширення (тип) імені файлу, як правило, вказує на те, до якого типу належить його зміст, а саме:

.**txt** – файл містить текст;

.**c –** у файлі міститься текст програми, написаний мовою СІ;

.**pas** –файл включає текст, написаний мовою ПАСКАЛЬ;

.**hlp** – у файлі міститься довідкова інформація (від англ. help – допомога).

Загалом, розширення, як і імена, можна задати довільно, проте певні програми працюють з файлами певного типу і щоб відрізнити їх від інших, слід дотримуватися загальноприйнятих назв, наприклад:

.**doc** – у файлі міститься текст, створений програмою WORD;

.**xls** – файл включає таблицю, створеною програмою EXCEL.

Каталог – являє собою спеціальне місце на диску, в якому зберігаються імена файлів, інформація про їхній розмір, відображено час їх останнього оновлення, атрибути файлів і т. д. Якщо каталог ***X*** зареєстровано у каталозі ***Y***, то перший вважають підкаталогом другого, а у той час ***Y*** – вважається надкаталогом або батьківським каталогом для ***X***. Каталог, у якому на даний момент працює користувач, називається поточним. Якщо потрібно звернутися до файлу, який розміщується не в поточному каталозі, треба визначити шлях до нього. Шлях – це певна послідовність імен каталогів чи символів “..”, розділених символом “\”. Цей шлях задає маршрут від поточного або кореневого каталогу на диску до того, в якому розміщено потрібний файл. Якщо шлях починається з символу “\”, то маршрут бере свій початок від кореневого каталогу, а в протилежному випаду від поточного, наприклад:

\DOS\LETTERS – шлях починається від кореневого каталогу;

C:\EXE\CHI – від поточного.

У заданні маршрутів можна застосовувати так звані маски, які створюються за допомогою символів ***** та **?** Символ ***** означає будь-яку кількість символів в імені файлу або в його розширенні. Символ ***?*** означає один довільний символ або відсутність символу в імені файлу (розширенні). Наприклад:

a:\work*.doc – усі ***doc***-файли із каталогу ***work*** (пошук, копіювання, вилучення та ін.);

****.bak*** – усі bak-файли у поточному каталозі.

Файли групуються в каталоги за цільовим призначенням, що значно полегшує їх пошук. Взагалі рекомендовано, щоб у кореневому каталозі перебувала мінімальна кількість файлів (це файли ОС, файли конфігурації, деякі драйвери) і підкаталогів. Це не тільки прискорює роботу з диском, але й полегшує орієнтування у файловій системі.

Атрибути файлів, тобто їхні властивості, призначено для різних потреб, а саме::

Read-only (тільки для читання) – цей атрибут захищає файл від змін та вилучення.

Hidden (прихований/ системний). Наприклад основні файли DOS IO.SYS і MSDOS.SYS, мають цей атрибут. Тому ці файли невидимі, не копіюються, не переміщуються в інше місце.

Archive (архівувати) – цей атрибут встановлюють для створення файлу, і при цьому програми архівного копіювання його відключають, як свідчення того, що копія файлу розміщена в архіві. Отже, наявність атрибуту "архівувати" означає, що для цього файлу не було створено архівної копії.

2.2.3. Команди MS DOS

Усі команди, за допомогою яких функціонує ОС, можна розподілити таким чином:

Сервісні команди:

cls – очистити екран;

time – вивести і змінити час;

date – вивести і змінити дату;

msd – вивести інформацію про комп'ютер;

Команда для роботи з диском:

c: – перейти на диск **C**:;

Команди для роботи з каталогами:

dir / p – вивести зміст каталогу сторінками;

cd <назва> – перейти в каталог з заданою назвою;

cd.. – повернутися в надкаталог;

cd – повернутися в кореневий каталог;

md <назва> – створити підкаталог з заданою назвою;

rd <назва> – вилучити порожній підкаталог;

ren <назва1> <нова назва> – перейменувати каталог **назва1**.

Команди для роботи з файлами:

copy <назва1> <назва2> – зробити копію файлу **назва1** і надати їй назву **назва2**;

copy *.* <шлях призначення> – скопіювати усі файли поточного каталогу за шляхом призначення;

copy <повний шлях звідки \ *.*> – скопіювати усі файли з деякого заданого каталогу у поточний;

copy <повний шлях звідки \ назва> – скопіювати заданий файл у поточний каталог;

copy con <назва> – створити текстовий файл з заданою назвою з клавіатури. Після введення тексту з клавіатури натиснути на F6 та клавішу введення;

copy <назва>prn – вивести файл на принтер;

copy <назва1> + <назва2> <назва3> – об'єднати два файли і дати їм назву **назва3**;

copy <назва1> + <назва2> – об'єднати два файли і дати їм назву **назва1**;

edit <назва> – викликати редактор Edit для редагування чи створення текстового файлу із заданою назвою;

type <назва> - вивести на екран текстовий файл;

del <назва> – вилучити текстовий файл;

move <назва> <шлях до каталогу призначення> – перемістити файл у заданий каталог;

ren <назва1> <нова назва> – перейменувати файл **назва1**.

2.3. Операційні системи WINDOWS

2.3.1. Файлова система та її структура в операційних системах WINDOWS

Інформація у комп' ютері зберігається в пам'яті або на різних носіях, наприклад, на жорстких дисках, або на компакт-дисках. Уся інформація, призначена для тривалого використання, зберігається у файлах. Для зручності файли зберігаються в різних папках (поняття "папка" в системі WINDOWS ідентично поняттю "каталог" в системі MS DOS), які розташовані на дисках. У комп'ютері може бути встановлено декілька дисків. Кожному диску присвоюється ім 'я у вигляді букви латинського алфавіту від **A** до **Z**, причому існують деякі правила позначення. Буквами **A** і **B** позначаються гнучки диски, буквою **C** - системний диск вашого комп'ютера, де розташована система Windows. Буквою **D** і подальшими буквами позначається решта дисків. Після букви, що позначає диск, ставитися символ ":", він показує, що буква позначає саме диск, наприклад **A**: або **C**:.

Кожен диск може вмістити безліч різних файлів. Кожний файл може розташовуватися як на самому диску, так і в будь-якій папці, що у свою чергу також може розташовуватися в іншій папці (див. рис. 2.1).

Встановлюючи шлях до файлу, імена папок відділяють одне від одного і від імені диска за допомогою символу зворотної косої риски "\", наприклад, **C:\ Мої документи\ Мої малюнки\ Ландшафт.jpg.** Даний запис означає, що файл з ім'ям **ландшафт.jpg** розташований у папці **Мої малюнки.** Ця папка у свою чергу перебуває в папці **Мої документи,** розміщеній на диску **C**:.

Важливим у сімействі операційних систем **Windows** є поняття ярлика. На будь-який об'єкт Windows можна послатися з іншого місця. Таке посилання і

називається ярликом. Наприклад, у якійсь папці розміщено малюнок, який часто використовується. Щоб забеспечити швидкий доступ до нього з різних місць, треба в кожному зробити за допомогою ярлика відповідні позначення, де буде відображено адресу фактичного місця зберігання цього малюнка

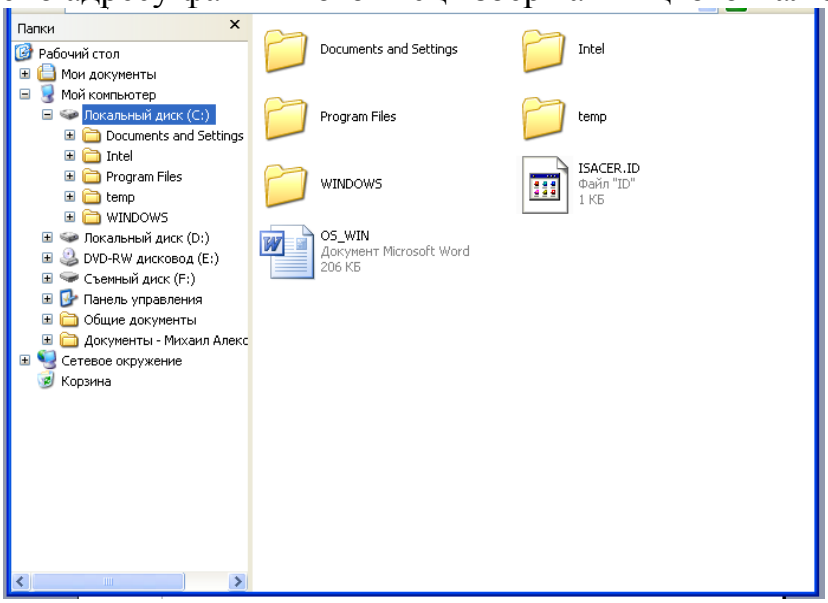


Рис 2.1. Структура зберігання інформації на дисках

2.3.2. Основні принципи роботи з системою

Після запуску операційної системи WINDOWS на всій площі екрана монітора з'являється особливий рисунок, який називається **Робочим столом** або, в англійській версії, **Desktop** (рис. 2.2). Діалогові вікна Windows розташовуються, накладаючись на робочій стіл, і їх можна переміщати, збільшувати, зменшувати або прибирати. Вікна можуть повністю або частково перекривати одне одного. Робочий стіл дозволяє запускати програми, налагоджувати систему і виконувати інші дії. У нижній частині робочого столу розташовано смугу, яка має назву **Панель задач**. Основне призначення цієї панелі – відображати програми, що запущені у вигляді кнопок і значків, забезпечити перемикання від одної до другої. Крім того, за допомогою панелі завдань можна запускати деякі необхідні програми.

У лівій частині панелі завдань знаходиться рамка, усередині якої розташовано слово **Пуск**. У системі **WINDOWS** такі рамки з текстом і рисунками всередині називають кнопками. Призначення цієї кнопки – запуск програм і налагодження роботи комп'ютера.

У системі **Windows** різних програми розпочинають роботу за допомогою кнопки **Пуск** або за допомогою значків на робочому столі, які називають ярликами. На робочому столі Windows також розташовано панель мови. Вона забезпечує швидке перемикання операційної системи для роботи різними мовами. Деякі додатки ОС, наприклад, програми, що входять до складу **Microsoft Office XP**, передбачають розміщення додаткових кнопок на панелі мови.

Рис 2.2. Загальний вигляд робочого столу **WINDOWS**

2.3.3. Головне меню WINDOWS

Головне меню (рис. 2.3) – це перелік програм і функцій, що може призначатися для швидкого доступу до всіх ресурсів ПК.

Головне меню дає змогу виконувати більшість завдань, пов'язаних з роботою операційної системи:

- запускати програму;
- відкривати найбільш уживані папки користувача;
- відкривати документи, з якими користувач працював останнім часом;
- проводити пошук даних;
- отримувати довідкову інформацію;
- проводити настроювання системи;
- виходити з системи, вимикати ПК.

Головне меню відкривають за допомогою кнопки ***Пуск*** або натисканням клавіші **WinKey** на клавіатурі.

У верхній частині меню зазначається ім'я користувача, який працює в цей момент на комп'ютері. Поряд з ім'ям розміщено малюнок, пов'язаний з цим користувачем.

Центральна частина меню розділена на дві колонки. Ліворуч розміщено команди, призначені для запуску постійно діючих програм. Праворуч у центральній частині розташовано команди, яки дозволяють переглядати вмісту важливих папок комп'ютера.

Рис. 2.3. Загальний вигляд головного меню **WINDOWS**

Команди, розташовані нижче, призначені для запуску програм, з якими останнім часом працював користувач. Як тільки ви запустите будь-яку програму, відповідна команда з'явиться саме в цій частині головного меню. При цьому програми, що давно не запускались, можуть бути вилучені з меню.

Нижче розташовано команду ***Все программы,*** яка відкриває доступ до програм, встановлених на ПК.

У нижній частині меню містяться команди для завершення роботи **WINDOWS**. За їхньою допомогою можна призупинити роботу комп'ютера, перезавантажити його або вимикнути (***Выключение***). Крім того, можна ввійти в систему під іншим іменем (**Выход из системы**).

Налагодження головного меню здійснюється з діалогового вікна ***Настройка меню «Пуск»*** (рис. 2.4), що викликається командою ***Пуск→Панель управления→Панель задач и меню «Пуск»***, кнопка ***Настроить***.

Використовуючи це діалогове вікно, можна настроїти:

- вигляд значків, що відображатимуться в меню: ***Крупные значки*** або ***Мелкие значки***;
- задати кількість програм, що відображуються в списку найчастіше вживаних програм за допомогою кнопки ***Количество программ в меню «Пуск».***
- вилучення ярликів із списку програм, які найчастіше використовуються кнопкою ***Очистить список***. При цьому, як правило, програми з комп'ютеру не видаляються;

● програми, які відображуються в меню ***Пуск*** для виходу в Інтернет і читання електронної пошти.

2.3.4. Контекстне меню

Контекстне меню з'являється на екрані після натискання правої кнопки миші у місці об'єкта. Воно відображує доступ до цього об' єкта саме в цей момент. Щоб викликати контекстне меню для виділеного елемента за допомогою клавіатури, потрібно натиснути комбінацію клавіш ***Shift + F10***. Набір команд у меню залежить від об’єкта, стасовно якого викликається контекстне меню. Натискання правої кнопки миші на значок або на ім'я файлу відкриває меню, в яке входять такі команди: ***Открыть, Копировать, Удалить*** і ***Переименовать***.

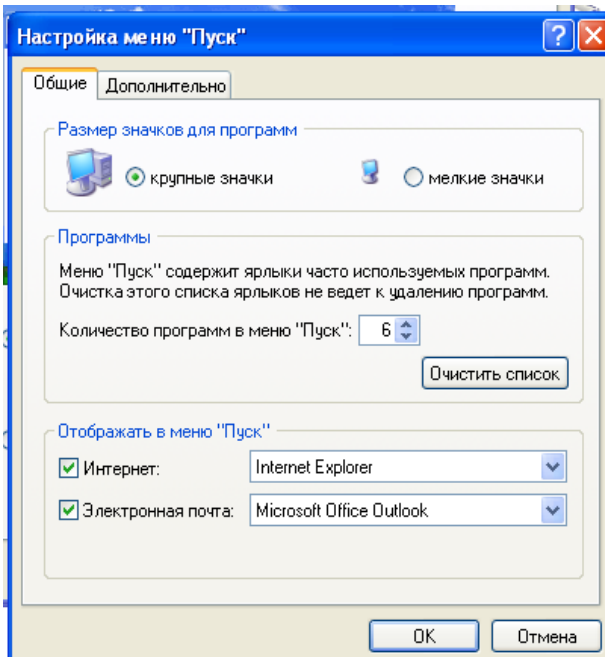


Рис. 2.4. Вікно настроювання головного меню

2.3.5. Завершення роботи з комп'ютером

Раптове вимкнення живлення комп'ютера під час роботи **WINDOWS** може зіпсувати важливу системну інформацію і призвести до непрацездатності системи. Тому просто вимикати комп'ютер з електричної мережі під час роботи з Windows не рекомендується. Для правильного завершення роботи із системою **WINDOWS** потрібно скористатися командою головного меню ***Выключение***. На екрані з’явиться діалогове вікно ***Завершение работы Windows,*** яке пропонує зробити вибір з варіантів для подальшиз дій, натиснувши одну з кнопок у центрі вікна. Якщо натиснути кнопку **Отмена**, вікно зникне з екрана і виконується повернення у звичайний режим роботи операційної системі **WINDOWS**.

Для завершення роботи системи варто натиснути кнопку ***Выключение***, унаслідок чого почнеться поступовий процес припинення функціонування роботи з **WINDOWS**, що може тривати кілька хвилин. Після закінчення цього процесу комп'ютер вимикається автоматично. Коли деякі програми операційної системи почали працювати ненадійно або перестали відповідати на запити користувача, то варто натиснути кнопку **Перезагрузка**, що зумовить перезапуск **WINDOWS** без вимкнення комп'ютера. При натисканні кнопки **Ждущий режи**м робота програм призупиняється, вимикаються монітор, диски, і ПК “засинає”. При поновленні роботи усі системи ПК, які діяли до переходу в режим очікування, повністю відновлюється.

2.3.6. Дії системи WINDOWS у разі виникнення збоїв

WINDOWS постійно контролює роботу всіх запущених додатків, перевіряючи стан іхньої роботи. Коли додаток перестає відповідати на запити, Windows виводить на екран діалогове вікно ***Завершение программы.***

У цьому вікні можна натиснути кнопку ***Завершить сейчас***, щоб припинити роботу програми. При цьому будуть втрачені зміни, що були внесені цим додатком у дані, починаючи з моменту останнього збереження.

Якщо програма перестає реагувати на команди користувача, то можна викликати діалогове вікно ***Диспетчер задач Windows*** (рис. 2.5) натискуванням такої комбінації клавіш ***<Ctrl>+<Alt>+<Del>,*** або викликати контекстне меню на кнопці цього додатка, яка розташована на панелі завдань, і натиснути кнопку ***Снять задачу.***

2.3.7. Робота з вікнами, вікна і діалоги

Вікно на екрані – це обмежена прямокутною рамкою поверхня екрана. У ньому відображається назва працюючої програми або документа.

Для налогодження системи використовуються діалогові вікна, яки мають стандартний зовнішній вигляд. Уніфікація елементів вікон скорочує час, який витрачається на їх вивчення.

Розрізняють три варіанти розміру вікна, яке відображується на екрані:

● ***стандартне*** – займає певну частину площини екрана, при потребі таке вікно можна перемістити в інше місце екрана;

● ***розгорнуте на весь екран (повноекранне)*** – займає всю площу екрана, а тобто максимальний розмір, його не можна переміщувати;

● ***згорнуте в піктограмму*** – зображується у вигляді кнопки на панелі завдань, при цьому виконання програми не припиняється, а щоб відкрити згорнуте вікно або згорнути вже відкрите, слід натиснути кнопку вікна на панелі завдань.

Зменшити розмір вікна до стандартного або розгорнути його в повноекранний режим можна подвійним клацанням миші у місці заголовка вікна.

Для прикладу розглянемо типове вікно програми **WordPad**, призначеної для створення і редагування текстових документів. Для запуску програми слід вибрати команду головного меню: **Программы → Стандартные → WordPad**. Програма буде запущена, і на робочому столі з'явиться її вікно. Розглянемо основні елементи вікна програми (рис. 2.6).

Кожне вікно програми має заголовок, в якому зазвичай відображаються назва програми і назва редагованого документа, наприклад, **Документ WordPad.**

У лівій частині заголовка розташовано значок програми, а у правій – керуючі кнопки, а саме:

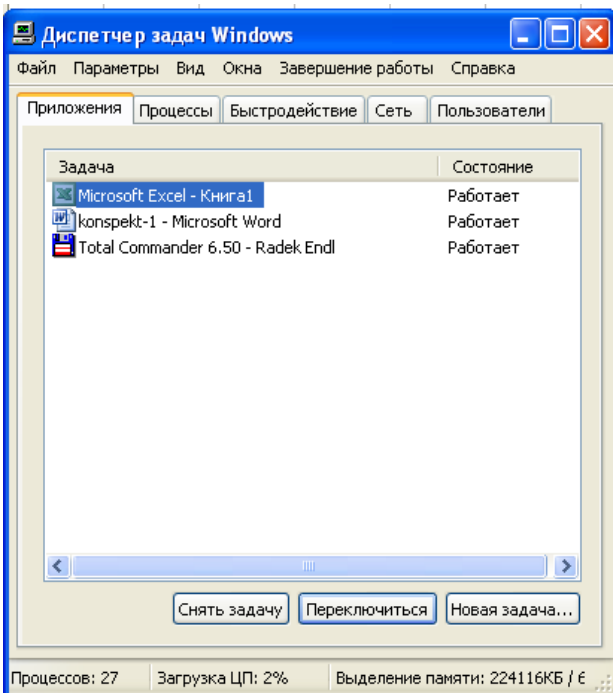


Рис. 2.5. Діалогове вікно **Диспетчер задач Windows**

- кнопка  ***Свернуть*** згортає вікна, але запущена програма у згорнутому вікні продовжує виконуватися;

- на кнопці  ***Развернуть*** у вікні стандартного розміру зображено квадрат, верхня межа якого прокреслена товстою лінією. Після натискання мишею у місці цієї кнопки вікно розкривається на весь екран;

- кнопка  ***Восстановить обратно*** дає змогу поновити попередні розміри вікна. У розгорнутому на весь екран вікні вона займає місце кнопки ***Развернуть***. Натискання кнопки змінює повноекраний розмір вікна на стандартний. Поновити або розгорнути вікно можна також подвійним клацанням миші у місці смуги заголовка;

- кнопка  ***Закрыть*** закриває вікно і завершує роботу програми.

Під заголовком розташовується так зване меню. Меню дає можливість вибирати різні дії за допомогою команд. Та чи інша команда перелічених у меню може бути вибрана за допомогою миші або клавіатури. Оскільки, як правило в програмі буває багато команд, то вони не вміщуються в одному рядку, і їх розташовують у багатьох вкладених меню. Наприклад, при роботі в графічному редакторові **Paint**, команда меню **Вид** приховує цілу групу команд, які також можуть викликати ще одну групу команд. Робота з меню будь-яких програм не відрізняється від роботи з головним меню **WINDOWS**. За допомогою миші або клавіатури ви послідовно вибираєте потрібні команди меню, відкриваючи при необхідності допоміжні меню (рис. 2.7). Надалі, при

описі будь-яких дій послідовність слів **Вид → Масштаб → Другой** означатиме необхідність поетапного вибору трьох команд меню. Назва деяких команд в меню можуть бути блідими, ніж інші. Це означає, що ці команди в даний момент недоступні. Наприклад, ви не можете вибрати звичайний масштаб перегляду, якщо вже працюєте в цьому режимі. За допомогою меню можна виконати будь-які дії, проте існують також інші способи роботи з програмами.

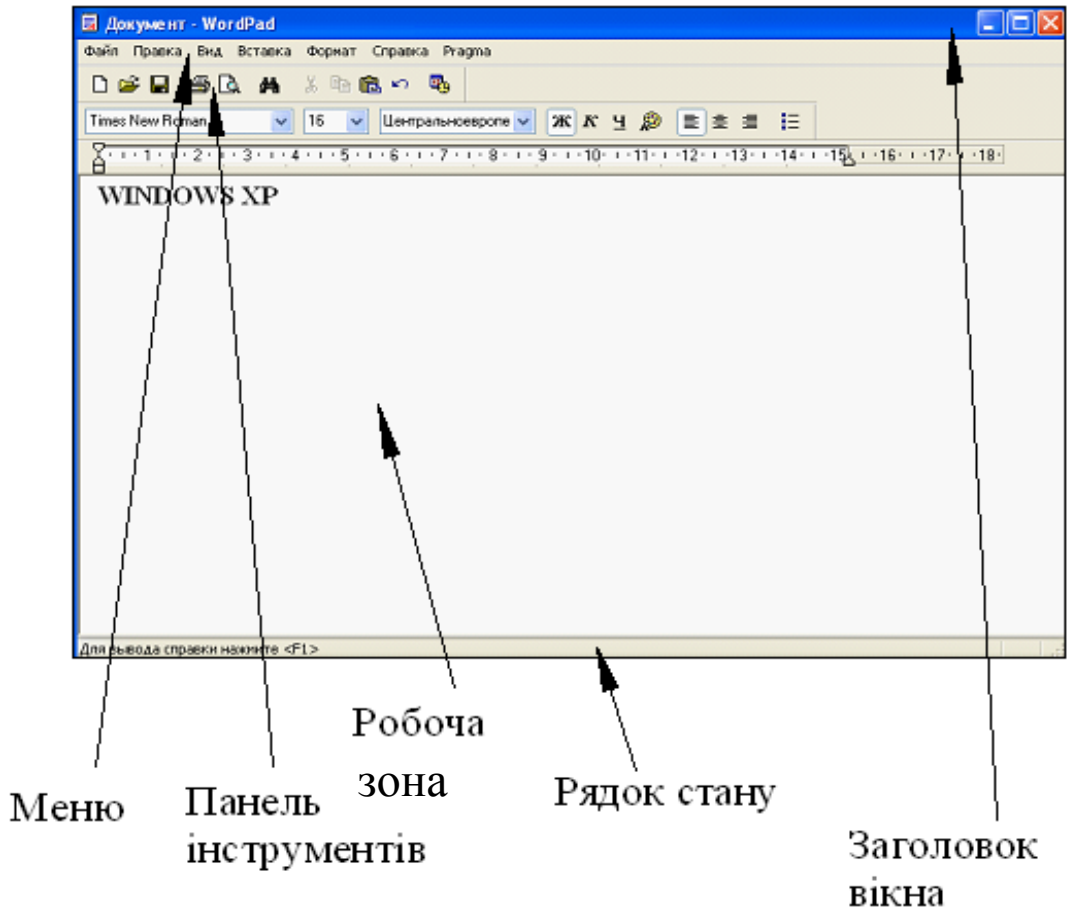


Рис. 2.6. Загальний вигляд типовогоробочого вікна програми

У середині вікна програми розташоване робоче поле програми. У нижній частині вікна розміщено рядок стану. Щоб відобразити його на екрані, потрібно вибрати в меню ***Вид*** команду ***Строка состояния***. Рядок стану складається з кількох ділянок, які містять інформацію, пов'язану з поточними діями користувача (кількість виділених об'єктів та їх розмір, призначення команди меню, на якій встановлено покажчик тощо).

Перегляд елементів тексту один за одним шляхом прокручування є електронним еквівалентом читання скрученого в рулон документа, на відміну від перегортання сторінок книги. Вертикальна та горизонтальна смуги прокручування автоматично з'являються вздовж правої межі і внизу вікна тоді, коли весь його вміст повністю не відображується. На кінцях смуг розташовуються дві кнопки, а між ними – бігунок.

Переміщуючи бігунок вертикальної смуги прокручування вгору або вниз при натиснутій кнопці миші, можна швидко відшукати потрібну інформацію у вікні. Натискання мишею у місці кнопки, розташованої на кінці смуги

прокручування, переміщує текст документа на один рядок. Натискання смуги прокручування зміщує зображення на одне вікно.

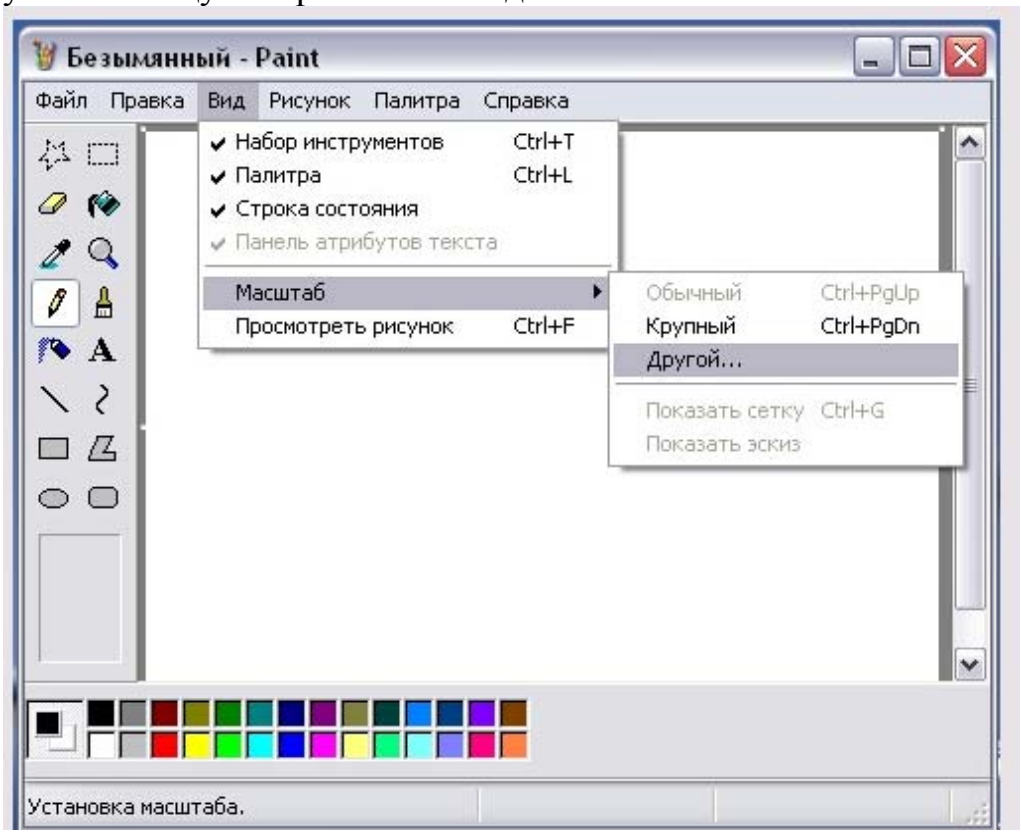


Рис. 2.7. Вкладене меню графічного редактора **Paint**

За положенням і довжиною бігунка можна визначити, яку частину вмісту вікна видно на екрані. Якщо на вертикальній смузі прокручування він зміщений на ¼ вниз, то ¼ вмісту вікна розташовано вище від його верхньоъ межі. За розміром бігунка відносно довжини смуги можна визначити, яку частину вмісту зображено у вікні.

Для переміщення вікна покажчик встановлюють на заголовку і, утримуючи натиснутою ліву кнопку миші, вікно переміщують у потрібне місце робочого столу.

Вікно, яке займає весь екран, не переміщується. Кожне вікно запам’ятовує свій розмір і стан на екрані; при повторному відкриванні з’явиться саме в тому стані, в якому перебувало перед закриттям.

Якщо ви відкриєте кілька вікон, то вони можуть розташовуватися на екрані поряд одне з одним, на деякій відстані, частково або повністю перекриваючи одне одного.

Вікно, з яким працює користувач у даний час, називають активним. Воно розташовується на передньому плані поверх інших вікон. Його заголовок відрізняється кольором або текстурою символів.

Під час роботи з кількома вікнами виникає потреба у переході від одного вікна до іншого. Найпростіший спосіб переходу в інше вікно – натиснути будь-яку видиму його частину. Якщо на екрані одночасно видно не всі вікна, то перехід до іншого вікна можна виконати одним із таких способів:

- натиснути мишею на панелі завдань кнопку з назвою потрібного вікна;

● одночасно натискуючи на клавіші ***<Alt>*** + ***<Tab>***, можна “по колу” переглянути всі працюючі програми, якщо натиснути***<Alt>*** +***<Shift>*** + ***<Tab>,*** то працюючі програми відображуються у зворотному напрямку і після вибору потрібного значка слід відпустити натиснуті клавіші;

● натиснути на комбінацію клавіш ***<Alt>*** + ***<Esc***> тоді активним стане наступне відкрите вікно.

Розташування кількох відкритих вікон можна впорядкувати на екрані за допомогою контекстного меню, яке відображається після натискування правою кнопкою миші на вільному місці панелі завдань. Контекстне меню містить команди, яки дають змогу впорядкувати розташування вікон в одному з таких положень:

● ***Окна каскадом*** – відкриті вікна розтошовуються каскадом одне над одним, перекриваючись;

● ***Окна сверху вниз*** – відкриті вікна розміщуються одне над одним, не перекриваючись, в один або кілька рядів;

● ***Окна слева направо*** – відкриті вікна шикуються в один горизонтальний ряд без проміжку або перекривання.

Слід враховувати, що користуючись командами меню, можна впорядкувати тільки незгорнуті вікна. Згорнуті вікна на екрані не відображуються.

Щоб згорнути всі відкриті вікна, слід скористатися кнопкою ***Свернуть все окна*** на панелі швидкого запуску. Відкриті діалогові вікна ця команда згорнути не може.

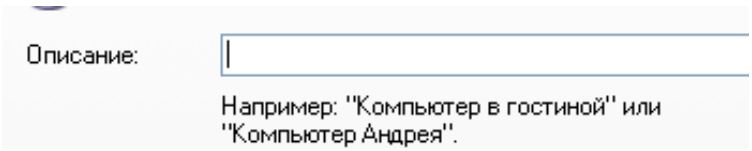
2.3.8. Діалогове вікно та його основні елементи

Діалогові вікна використовують для наголодження режимів роботи операційної системи, обладнання, програми тощо (рис. 2.4).

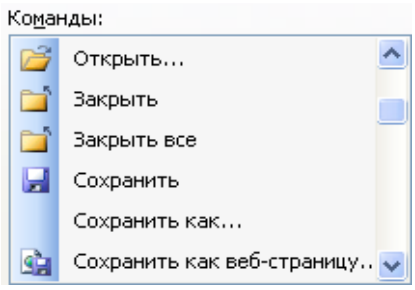
Як правило, у діалоговому вікні подано кілька вкладок. Кожна вкладка має ярличок з написом, розташований нижче від смуги заголовка. Натискання кнопкою миші на ярличок виводить вкладку на передній план (тобто відкриває її) і дає змогу працювати із зображеними на ній елементами. У діалоговому вікні відображується набір елементів керування: кнопок, прапорців, перемикачів тощо.

Діалогове вікно зазвичай складається з таких елементів:

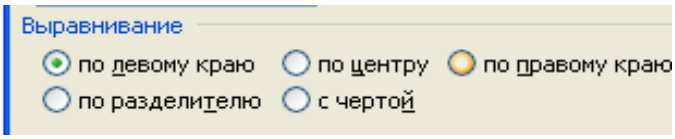
● ***текстове поле*** – прямокутна ділянка, куди можна ввести текстову інформацію за допомогою клавіатури;



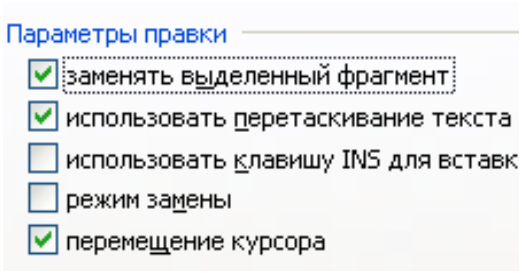
● ***список, що розкривається*** – має вигляд текстового поля, доповненого в правій частині кнопкою розкриття із стрілкою, спрямовоною вниз;



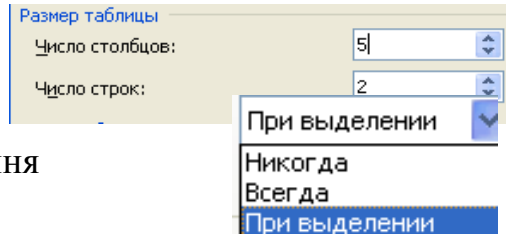
● ***перемикач*** - використовують для вибору одного з кількох параметрів, які взаємно виключають один одного;



● ***прапорець*** – використовують для вмикання або вимикання однієї чи кількох функцій. Він має вигляд маленького квадрата. Коли прапорець встановлено, у квадраті видно помітку, коли прапорець відсутний, то квадрат пустий. Якщо прапорець у цей момент для користувача недоступний, то квадрат замальовується сірим кольором;

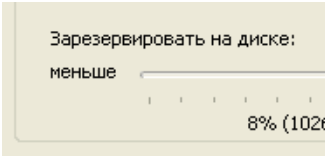


● ***лічильник*** – використовують для зміни числового значення параметра за допомогою кнопок-стрілок, розташованих справа від поля;

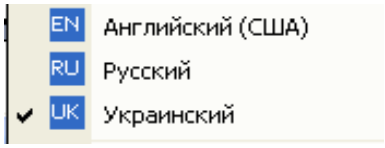


● ***список*** – містить набір запропонованих елементів, а поточне встановлення елемента виділяється кольором або інверсією;

● ***регулятор (бігунок)*** – застосовують для плавної зміни режимів роботи пристрою, тому переміщуючи бігунок, ми змінюємо значення заданого параметра;



● ***індикатор*** – відображує стан пристрою, наприклад, індикатор мови показує, на яку мову налаштовано клавіатуру;



● ***індикатор ходу роботи*** – відображує в діалоговому вікні хід виконання поточної операції, наприклад, за його допомогою можна стежити за виконанням операції копіювання;

● ***кнопки команди*** мають вигляд прямокутників з написом, при цьому натиснення кнопкою миші в місцях цих прямокутників зумовлює виконання

або відмову від виконання певних операцій, для цього найчастіше використовують кнопки ***ОК, Отмена, Применить;***

● кнопка ***Применить*** дає змогу зберегти виконані установлення, не закриваючи діалогове вікно, а натиснення мишею кнопки ***ОК*** означає виконання встановлених налаштувань і закриття діалогового вікна, а використання кнопки ***Отмена*** закриває діалогове вікно без збереження внесених змін (рис. 2.8.).

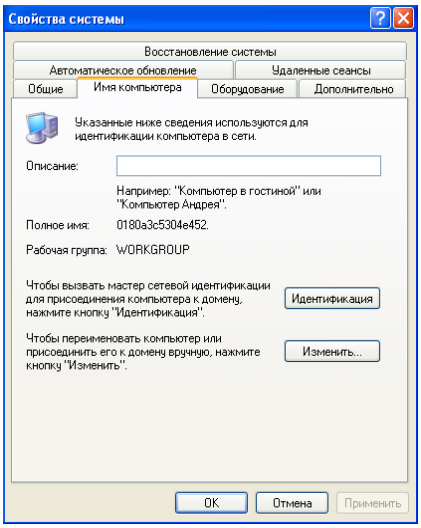


Рис. 2.8. Приклад загального вигляду діалогового вікна

2.4. Провідник в операційній системі WINDOWS

Програма **Проводник** призначена для навігації (подорожі) файловою структурою і для виконання дій з її об’єктами: копіювання, переміщення, перейменування, відшукування файлів і папок тощо.

Для запуску провідника слід вибрати команду головного меню Windows **Другие программы → Стандартные → Проводник**. Після запуску програми на робочому столі з'явиться її вікно (рис. 2.9), зовнішній вигляд якого може бути дуже різним, залежно від налаштувань програми.

Провідник можна викликати також іншим способом, наприклад, натиснувши лівою клавішею миші в місці значка **Мои документы** або значка **Мой компьютер** на робочому столі **WINDOWS**. Також є можливість вибрати аналогічні команди в головному меню **WINDOWS**.

Вікно провідника складається з двох головних частин: дерева папок ліворуч і робочого поля зі змістом активної папки праворуч. Додатково можна увімкнути панель інструментів і рядок статусу.

Панель інструментів користувач може налаштовувати на свій розсуд, додаючи за допомогою команди ***Вид*** такі нові кнопки:

● ***эскизы страниц*** – при перегляді папок показує у вікні мініатюрне зображення вмісту графічних файлів, веб-сторінок;

● ***плитка*** – показує над іменами папок і файлів великі значки;

● ***значки*** – відображує, крім імен папок і файлів малі значки;

● ***список*** – сортує об’єкти у вікні за їх назвами в алфавітному порядку,

при цьому спочатку виводяться імена папок, потім файлів, а значки об’єктів розміщуються в один або кілька стовпчиків поряд з їх назвами;

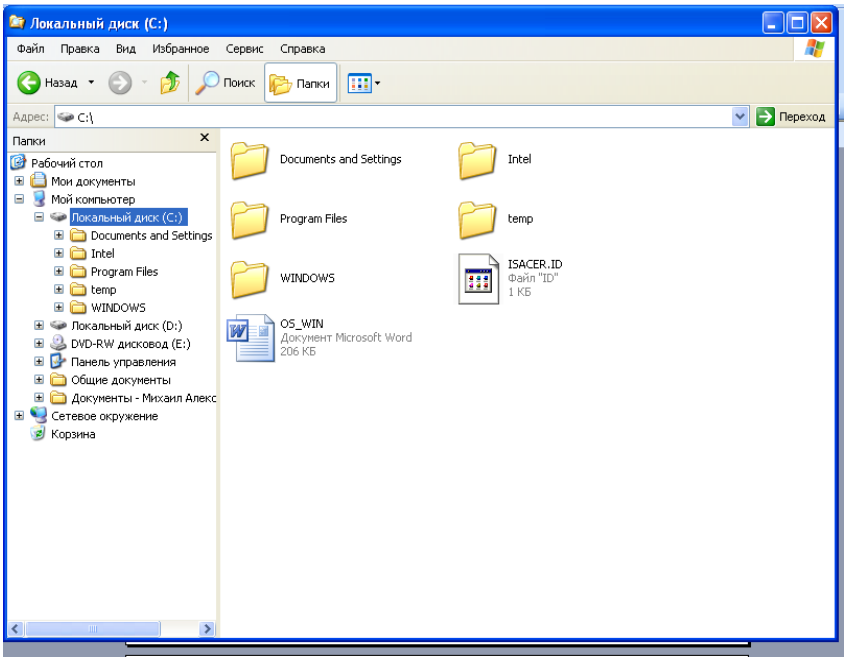


Рис. 2.9. **Загальний вигляд р**обочего вікна провідника Windows XP

● ***таблица*** – подає список об’єктів з такими даними про них: ***Имя, Размер, Тип, Дата изменения*** та ін. Це потрібно для сортування в алфавітному порядку папок та файлів, запропонованих у стовпчику таблиці. А щоб воно відбулось необхідно натиснути лівою клавішею миші у місцях заголовків цих папок і файлів. Сортування в зворотному порядку вимагає повтору операції.

Кнопки панелі інструментів ***Ссылки*** можна використовувати для відкривання часто відвідуваних веб-сайтів, а також тих папок і файлів, з якими ви постійно працюєте.

2.4.1. Методи роботи з дисками і папками

Якщо відомо, де розташований необхідний файл, можна детально послідовно ввести повний шлях до нього в полі введення **Адрес**. При цьому буде здійснено безпосередній перехід до папки, де міститься цей файл. Якщо для введення адреси використовують клавіатуру, то для переходу до потрібної папки, слід натискувати клавішу <Enter>.

Якщо працюють з провідником операційної системи Windows, то всі диски і папки комп'ютера представляються у вигляді дерева (рис. 2.9). В основі дерева, яке часто називають коренем, розміщено особливу папку, ім'я якої **Рабочий стол**. У цій папці розміщено службові папки, всі диски комп'ютера та інші важливі папки.

Коли ж якась папка містить інші папки, то ліворуч від її значка буде розташовано рамку із відповідною поміткою, в іншому випадку такої рамки

немає. При цьому вміст папки, поміченої рамкою, у списку не розкривається. Якщо ж ви бачите значок, то вміст папки відображено нижче в списку.

Зверніть увагу, що в провіднику ліворуч в зоні вікна відображаються тільки диски і папки, а окремі файли ні. Для цього призначено права частина робочого столу. Якщо клацнути мишею у місці папки, розміщеної зліва, то її вміст відобразитися в правому полі у вигляді значків. Відкрити вміст папки або диска можна також іншим способом. Якщо двічі клацнути мишею у місці значка папки, розміщеної в правому полі, то папка буде відкрита. Цей спосіб зручний для режиму не відображення списку. Послідовно виконуючи подвійні клацання у місцях потрібних папок, можна перейти вниз їхнього ієрархічного переліку. Коли ж потрібно виконати протилежну дію, то слід скористатися кнопкою на панелі інструментів, натиснувши яку, можна перемістити на один рівень вгору в ієрархії вкладених папок.

Треба зазначити, що всі пересування серед папок в рамках поточного сеансу роботи з провідником відображаються в пам'яті комп'ютера, тому можна в будь-який момент повернутися назад до папки або диска, переглянутих раніше, натискаючи кнопки  ***Назад*** і  ***Вперед***, які розташовані на панелі інструментів.

2.4.2. Копіювання, переміщення і перейменування файлів

Часто виникає необхідність в переміщенні документів з однієї папки в іншу. або, наприклад, потрібно розташувати цілком якусь папку в іншому місці.

Перш за все, зауважимо, що файли і папки на комп'ютері можна як переміщувати, так і копіювати. Операції копіювання, переміщення і видалення виконуються абсолютно однаково як для файлів, так і для папок з файлами.

Рис. 2.10. Діалог копіювання файлу

За наявності списку папок у лівій частині вікна програми, це завдання виконується дуже просто. Вибираємо початковий файл у правій частині вікна й перетягуємо його на нове місце, яке вибирається в списку всіх папок. Визначивши це місце, підводимо курсор миші до значка файлу в правій частині вікна, далі натискуємо кнопку миші і, не відпускаючи її, переміщуємо покажчик на вибране місце у лівій частині вікна, після чого відпускаємо клавішу. При переміщенні виділеного значка на місце, де розташований список, папка призначення також буде виділена, а відпускання кнопки миші означає початок виконання відповідної операції. Зауважимо, що коли перемістити файл у вибране місце призначення неможливе, то в момент перетягування курсор миші змінить свою форму. Під час копіювання та переміщення файлів може з'явитись діалогове

вікно, що ілюструє цей процес (рис. 2.10). Якщо копіюють невеликий файл, то вікно дуже швидко зникне або навіть зовсім не з'явиться. У роботі з великими файлами поступове збільшення смужки в нижній частині вікна відображає відсоток виконання операції. Натиснувши кнопку **Отмена**, операцію можна припинити. Коли процедура копіювання або переміщення буде завершена, діалогове вікно закриється.

Якщо в списку розміщено дуже багато папок і дисків, то місце призначення може бути невидимим. У такому разі потрібно перетягнути значок у верхню або нижню частину списку і почекати, не відпускаючи при цьому кнопку миші. Через деякий час список почне зміщуватися у вікні, і користувач побачить потрібну папку. Коли ж потрібно скопіювати файл у вкладену папку, яка не видна в списку, слід поступити так само.

Коли виникає потреба не переміщувати, а скопіювати файл у папку на тому ж самому диску, то, переміщуючи відповідний значок, необхідно натиснути й утримувати клавішу <**Ctrl**> на клавіатурі комп'ютера.

У разі переміщення значка в папку, розміщену на іншому диску, відповідний файл буде скопійований, а щоб його перемістити в папку на іншому диску, необхідно утримувати натиснутою клавішу <**Ctrl**>.

Зауважимо, що можна виконувати операції копіювання або переміщення не одного, а групи файлів чи папок. При цьому слід спочатку виділити групу значків, а потім перетягнути їх в потрібне місце. За необхідності виділення групи значків, розташованих поряд, слід натискувати й утримувати клавішу <**Shift**> на клавіатурі. Далі клацають лівою клавішею миші у місцях першого й останнього значка групи, після чого відпускають клавішу, і всі значки, розташовані у цьому проміжку, будуть виділені, а відповідні групи файлів і папок готови до переміщення в потрібне місце списку лівої частини вікна програми.

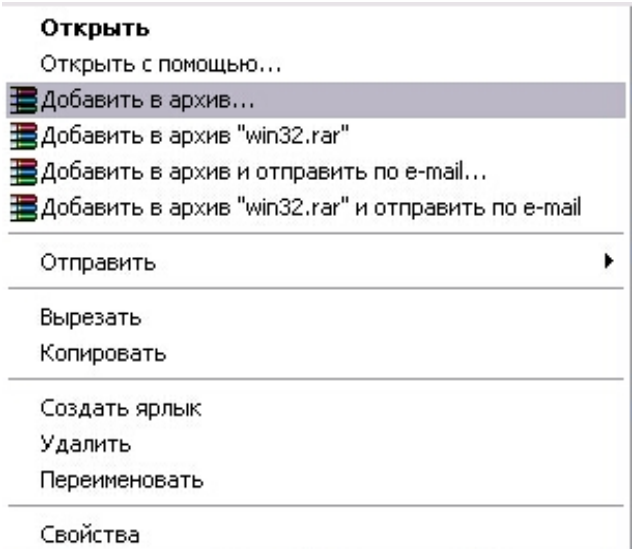


Рис. 2.11. Загальний вигляд допоміжного меню

Якщо клацнути правою кнопкою миші у місці певного значка або групи виділених значків, то поряд з'явиться допоміжне меню (рис. 2.11).

Кількість команд у меню і їх склад залежить від встановлених на комп'ютер програм, проте команди копіювання, видалення і перейменування будуть в цьому меню завжди. Щоб скопіювати файл або групу файлів, необхідно вибрати команду **Копировать** даного меню, потім необхідно перейти в папку призначення і знову відобразити допоміжне меню шляхом натискання правої кнопки миші у вільному місці робочої зони програми. У меню потрібно вибрати команду **Вставить**, щоб відбулося копіювання файлів.

При такому способі для тимчасового зберігання переміщуваних і копійованих файлів служить так званий буфер обміну **WINDOWS**. З метою переміщення файлу потрібно, замість команди **Копировать** допоміжного меню, вибрати команду **Вырезать**. При цьому файл буде видалений з поточної папки і поміщений у буфер обміну **WINDOWS**. Якщо після цього файл не буде вставлений в іншу папку, то його втрачено.

Допоміжне меню передбачає ще ряд корисних команд, наприклад команду **Переименовать**, при використанні якої, замість імені вибраного значка з'являється поле введення, в якому за допомогою клавіатури можна ввести його нове ім'я на свій розсуд. Введення імені закінчується натисненням відповідної клавіші на клавіатурі комп'ютера.

Контрольні питання

1. Дайте визначення операційної системи комп'ютера, для чого вона призначена?
2. Які основні функції виконує операційна система комп'ютера?
3. Що собою являє ім'я файлу і його розширення (тип)?
4. Що таке повне ім'я файлу?
5. Що таке атрибути файлу?
6. Яке призначення об'єкта «***Мой компьютер***»?
7. Яке призначення панелі задач?
8. Які дії можна виконувати за допомогою діалогового вікна?
9. Яким чином можна зробити вікно активним?
10. З якою метою використовують контекстне меню?
11. Яким способом можна відкрити папку?
12. Як відбувається правильне вимикання комп'ютера?
13. Що таке поняття повний шлях до файлу?
14. Які символи не використовуються в назвах файлів?
15. Яке призначення папки?
16. Які дії визначені над папками?
17. Як створити ярлик?
18. Які дії визначені над ярликами?
19. Які дії визначені над файлами?
20. Яке призначення контекстного меню об'єкта?
21. Як створити папку на диску?
22. Як перемкнути мову на клавіатурі?
23. Як відбувається перейменування робочої папки?
24. Як увімкнути панель інструментів?
25. Як перемістити папки на інший диск?
26. Як вилучити файл?
27. Що таке копіювання об'єктів і як його виконують?
28. Яка відмінність між переміщенням і копіюванням об'єктів?
29. Як скопіювати файл за допомогою буфера обміну?
30. Для чого призначений буфер обміну?

31. Яке призначення програми ***Проводник***?
32. Як запустити програму ***Проводник***?

3. АЛГОРИТМІЗАЦІЯ ТИПОВИХ ЗАДАЧ

3.1. Загальні положення

Етап алгоритмізації даних включає математичне формулювання задачі й розробку алгоритму її розв'язку.

Під алгоритмом варто розуміти точне приписання, що визначає обчислювальний процес, що веде від вхідних даних, що варіюються, до шуканого результату.

Відзначимо основні властивості алгоритмів:

● ***точність*** – передбачає встановлення чіткого порядку дій, щоб, виконавши чергову команду, виконавець точно знав, яку операцію треба виконувати далі;

● ***масовість*** – це можливість за допомогою одного алгоритму розв'язувати не тільки індивідуальну задачу, але й інши однотипні задачи на базі різних початкових даних;

● ***формальність*** — тобто можливість людини (програміста) правильно скласти програму за даним алгоритмом.

Отже програма являє собою запис алгоритму засобами мови, що сприймається ЕОМ, а також остаточний набір інструкцій, які інакше можна назвати командами. Тобто програму, за якою ЕОМ проводить обчислення відповідно заданому алгоритму, можна подати у вигляді послідовності інструкцій – команд, кожна з яких забезпечує виконання певної дії комп'ютера і фізичну реалізацію алгоритму розв'язку задачі.

Перш ніж скласти програму мовою машини або алгоритмічною мовою, необхідно описати алгоритм задачі. Основне призначення цього запису – полегшити подальший процес програмування. Для опису алгоритмів можна використовувати різні способи, характеристику яких подаємо нижче

3.2. Особливості мови графічних символів

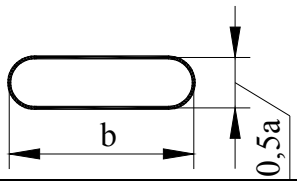
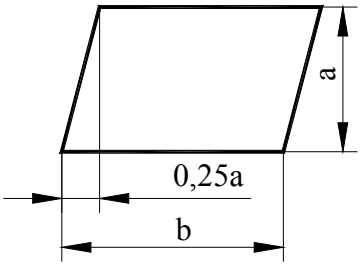
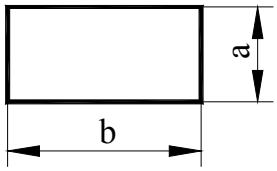
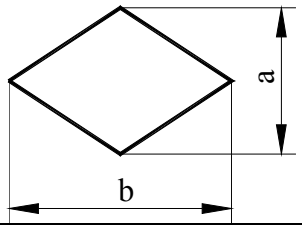
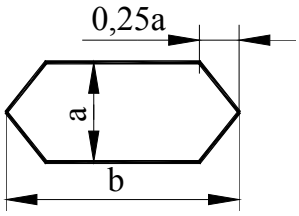
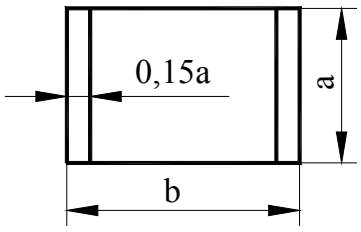
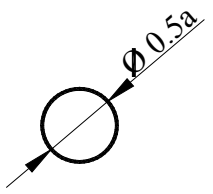
Для зображення структури алгоритмів використовується сукупність керуючих блокових символів (блоків), які з'єднуються лініями. Таке зображення називається схемою алгоритму. Оскільки алгоритми сприймаються насамперед візуально, то їхнє зображення повинно мати чітку й виразну структуру. Стислість, виразність і планомірність в роботі над створенням схеми алгоритмів дозволяють допомогтись їхньої високої якості.

У схемі алгоритму кожному типу дій (наприклад, введенню початкових даних, обчисленню значень виразів, перевірці умов, керуванню повторення дій, закінченню обробки і под.) відповідає геометрична фігура, яка являє собою блоковий символ, що прийнято називати символом дії. Символи дії з'єднуються між собою лініями переходів, що визначають черговість виконання операцій.

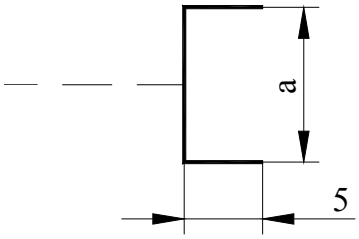
Найбільш часто використовувані символи схем алгоритмів наведено в таблиці 3.1.

Таблиця 3.1

Основні символи схем алгоритмів

Назва	Позначення	Пояснення
Пуск, зупинка		Початок, кінець, зупинка, вхід в підпрограмах і вихід з них
Введення, виведення		Перетворення даних у форму, придатну для обробки (введення) або відображення результатів обробки (виведення)
Процес		Обчислювальна дія або послідовність обчислювальних дій
Розв'язок		Перевірка умов
Модифікація		Початок циклу
Зумовлений процес		Обчислення за підпрограмою
З'єднувач		Розрив лінії потоку

Закінчення табл. 3.1

Коментар		Пояснення, зміст підпрограм, формули

Мінімальне значення розміру відрізка **a** дорівнює 10 мм, а його збільшення зазвичай відбувається на число, кратне 5. Розмір відрізка **b** приймається рівним **1,5a** (допускається також **b=2a).** У межах однієї схеми рекомендовано зображувати символи однакових розмірів. Контури символів і ліній потоків, що їх з'єднують, виконуються суцільною (безперервною) лінією, товщина якої становить 0,6 – 1,5 мм.

У середині символу описується зміст операції (або операцій).

Символ обмеження (пуск,зупинка) призначений для позначення входів у схему алгоритму й виходів із неї. Кожна схема повинна починатися, або закінчуватися символом обмеження. У цих символах дозволяється давати пояснення до використання. Якщо символ указує на переривання, то він повинен ідентифікувати відповідну виняткову ситуацію й схеми, що здійснює управління в цій ситуації. Охарактеризуємо коротко кожен із символів.

Символи введення-виведення інформації ***(введення, вивід)*** використовуються для позначення цих операцій. Окремим логічним пристроям ЕОМ або окремим функціям обміну відповідають певні блокові символи. У кожному з них зазначається тип пристрою або файлу даних, тип інформації, що бере участь в обміні, а також вид операції обміну.

Символ обробки (процес) застосовується для позначення однієї дії або послідовності дій, що змінюють значення, форму відображення або розміщення даних. Для поліпшення наочності схеми декілька окремих блоків обробки можна об'єднати в один блок. Відображення окремих операцій досить вільне, тобто не підлягає певним правилам. Наприклад, для позначення обчислень можна використовувати математичні вирази, для пересилання даних – стрілки, можливі також пояснення дії зрозумілою для комп'ютера мовою. Метод блок-схем, так само як і алгоритмічна мова (псевдокод), не залежить від специфіки мов програмування, тому в описах операторів не слід використовувати резервовані слова й символи із цих мов, а також застосовувати імена даних, утворені відповідно до їхніх синтаксичних правил.

Символ розв'язки (розв'язок) використовується для позначення послідовності переходу керування діями відповідно до певної умови. При цьому мають бути сформульовані питання, відповіді умовам або порівняння, які стосуються кожного етапу розв'язку**.** Стрілки, що виходять із блоку розв'язку, мають бути позначені потрібними відповідями (наприклад, **ТАК**, **НІ**), причому враховуються всі їхні можливі варіанти.

Символ модифікацій (модифікація) використовується для організації виконання циклічних конструкцій. Тоді в середині блоку записують певний параметр циклу, зокрема його початкове значення, граничну умову й правило зміни значення в кожному повторенні. Блок розміщується на початку циклічної конструкції, керування якою він здійснює, навіть у тому випадку, якщо зміна параметра та перевірка умов відбувається не на початку, а в кінці циклу.

З'єднувачі використовуються у тих ситуаціях, коли схема алгоритму розподіляється на автономні частини, зокрема якщо вона не вміщається на одному аркуші, або коли необхідно уникнути зайвих перетинів ліній переходів на зображеннях.

Коментар дозволяє включати в схеми алгоритмів пояснення до функціональних блоків. Часте використання коментарів небажане, оскільки це ускладнює (захаращує) схему, робить її менш наочною. Проте деякі позначення змінних, прийняті припущення або призначення окремих алгоритмів вимагають додаткових пояснень.

У даний час основна тенденція у використанні схем алгоритмів полягає не стільки у відображенні послідовності операцій, скільки в групуванні блокових символів як керуючих базових конструкцій. До таких відносяться проходження, розгалуження й повторення.

3.3. Алгоритми основних видів обчислювальних процесів

3.3.1. Загальні положення

Помічено, що алгоритм розв'язки будь-якої задачі можна уявити як комбінацію базових алгоритмів (обчислювальних процесів). До таких процесів належать:

- ***прості (лінійні) обчислювальні;***
- ***розгалужені обчислювальні;***
- ***циклічні обчислювальні.***

3.3.2. Простий (лінійний) нерозгалужений обчислювальний процес

Алгоритми простих нерозгалужених процесів не передбачають жодного етапу, який мав би більше більш одного спадкоємця, тобто алгоритми реалізуються шляхом виконання простої послідовності операцій.

Приклад 3.1. Записати алгоритм обчислення значення величини **Y** за такою формулою:

Y = 150 + a·x. Схема алгоритму подано на рис. 3.1.

3.3.3. Розгалужені обчислювальні процеси

Алгоритми розгалужених обчислювальних процесів включають принаймні один етап, який має більше одного спадкоємця. Необхідність уведення такого етапу зумовлюється виконанням певної умови. Іншими словами задача має

кілька шляхів розв'язку, які вибираються у залежності від виконання тієї чи іншої умови. Таким чином, алгоритм повинен не тільки описувати, але й вибирати потрібний хід розв'язку з урахуванням вхідних даних або проміжних результатів. Напрями, за якими може відбуватись обчислювальний процес, називаються його гілками.

Прикладам такого процесу є відбір і розрахунок вузлів і механізмів із різних деталей.

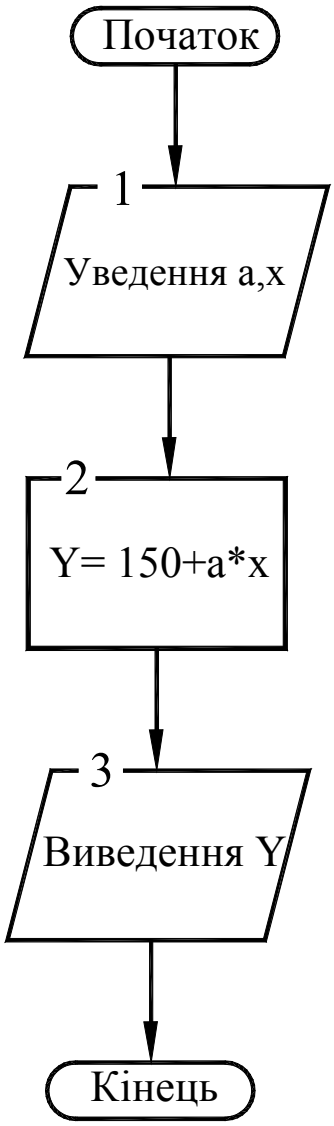


Рис. 3.1. Схема алгоритму простого (лінійного) процесу

Приклад 3.2. Записати у вигляді схеми такий алгоритм обчислення:

$$S=\begin{cases} \mathbf{kx + 2t}, \text{ якщо } \mathbf{t >= 0} \\ \mathbf{kx - 4t}, \text{ якщо } \mathbf{t < 0} \end{cases}$$

При записі алгоритмів розгалужених обчислювальних процесів необхідно дотримуватись таких вимог:

● у різних гілках можна використовувати й ті самі позначення змінних величин;

● обчислення або процеси, що повторюються в різних гілках схеми алгоритму, виносяться за межі розгалуження (у нашому прикладі це обчислення виразу **kx**), а потім в кожній з гілок виконується операція привласнення значенню **S** попереднього значення **S** і додаткових складових первинної формули, що визначає величину **S**;

● складні умови обчислень розбиваються на ряд простих.

Розв'язок задачі буде мати вигляд, поданий на рис. 3.2.

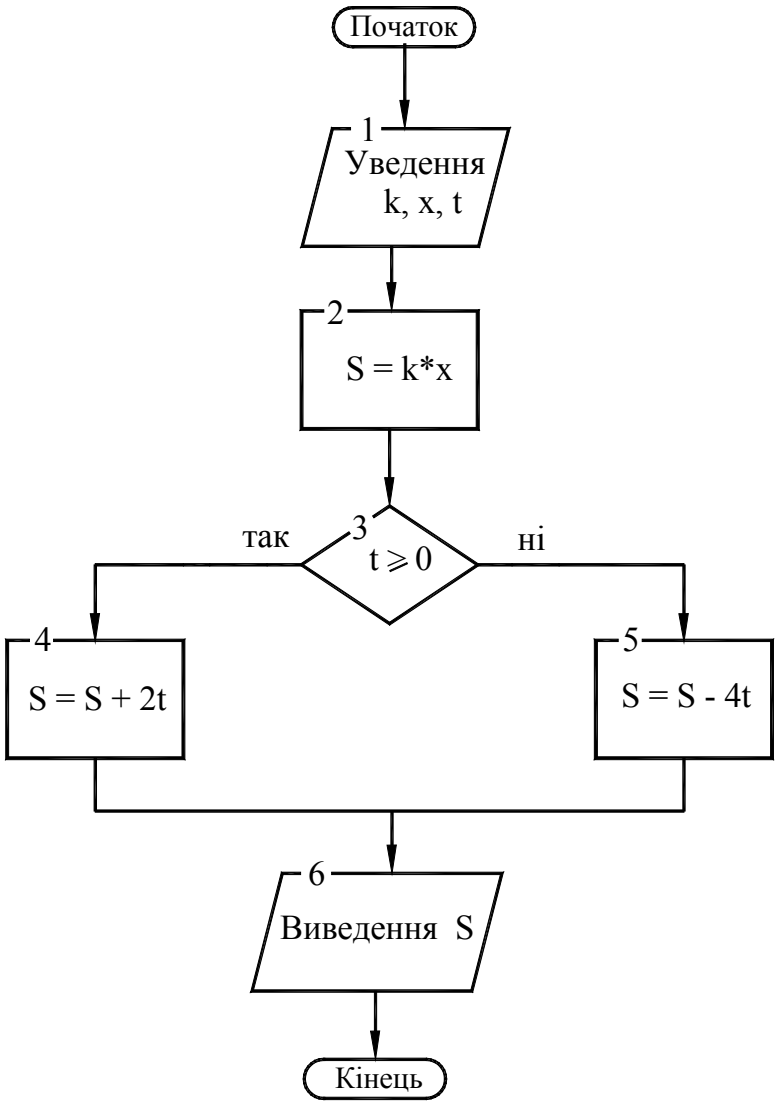


Рис. 3.2. Схема алгоритму простого розгалуженого обчислювального процесу

3.3.4. Циклічні обчислювальні процеси

Більшість практичних задач розв'язуються на ЕОМ за допомогою циклічних алгоритмів. У таких алгоритмах мають місце ділянки обчислень, що являють собою багаторазове повторення однієї і тієї самої послідовності операцій.

Наприклад, при обчисленні виразу y = $\mathbf{a^x}$, операція множення повторюється **x** раз. Якщо **x = 5** , то операція множення повторюється 5 разів, тобто y **= a · a · a · a · a ,** а коли **x = 100,** то множення повторюється 100 разів.

Таким чином, ***циклічний обчислювальний процес – це послідоність дій, яка передбачає повторювані n разів етапи обробки інформації.***

Цикл повинен тривати доти, поки не виконається деяка умова виходу з нього.

Приклад 3.3. Скласти схему алгоритму для обчислення такого виразу

$$y = \frac{x^3 - 4x + 1}{|x| + 1},$$

при цьому параметр циклу змінюється до $\mathbf{x_п}$ із кроком **h**. Такі обчислення значень заданої функції називаються табулюванням.

Виконуючи табулювання, необхідно послідовно перебрати всі значення параметра циклу **x** у зазначеному діапазоні його величин. Як тільки цю умова буде порушено, слід припинити обчислення. Схема алгоритму подається на рис. 3.3.

При уважному розгляді схеми алгоритму видно, що основою її розрахункової частини є робочі формули. Сморід й дозволяють обчислювати результат з огляду на задану кількість циклів. Виходить, що перш ніж зображувати блок-схему, необхідно скласти робочі формули. Пошук робочих формул для всіх типів циклів виконується за єдиною методикою. Зміст робочих формул дозволяє визначити послідовність розрахунків, методи трансформації змінних величин (параметрів) циклу. Змінна циклу при виході з нього набуває значення, яке постійне покроково нарощується до кінцевого значення.

Крім того, варто звернути увагу, що в перший символ уводяться окремі змінні, а кілька однотипних операцій уведення або виведення даних мають входити в цикл як багаторазово повторювані.

Циклічні процеси можна класифікувати залежно від кількості й складності циклів на ***прості (один цикл)*** і ***складні (цикл у циклі або розгалуження в циклі).*** У свою чергу **прості цикли** поділяються залежно від способу виходу із циклу на ***арифметичні*** й ***ітераційні.*** Розглянемо кожен з таких циклів.

3.3.5. Арифметичні цикли

Циклічний процес, у якому заздалегідь відомо кількість повторень циклу, називається арифметичним циклом.

Наприклад, розв'язок рівняння величини y = $\mathbf{a^x}$ при заданому значенні **x** або **y = n!** при заданому значенні **n,** реалізується за допомогою арифметичних циклічних процесів. Кількість повторень циклу в них задають величини **x** і **n** відповідно.

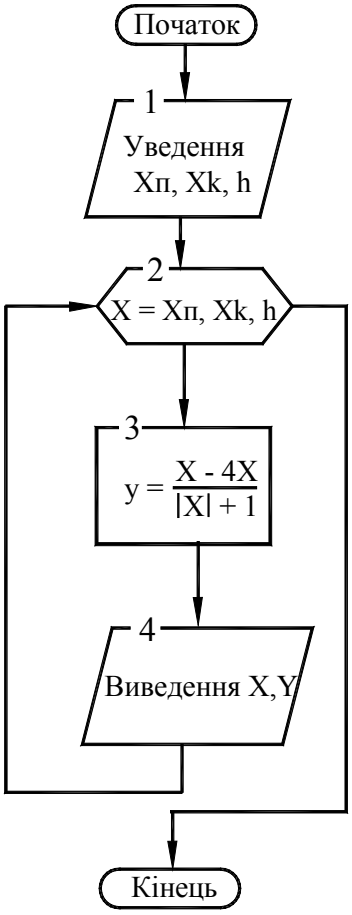


Рис. 3.3. Схема алгоритму простого циклічного обчислювального процесу

Для підрахунку кількості повторень кожного арифметичного циклу вводять спеціальні змінні, які називають ***лічильниками циклів.*** Лічильник являє собою цілу змінну величину. На підготовчому етапі обчислень у схемі алгоритмів необхідно задати початкове значення для лічильника. Якщо лічильник працює на збільшення, тобто початкове значення його становить 0 чи 1, а після кожного виконання циклу лічильник збільшується на одиницю, то такий лічильник називається ***прямим.*** Якщо початкове значення відповідає максимальному числу повторень циклу, а після кожного виконання циклу воно зменшується на одиницю, то такий лічильник називається ***оберненим.***

3.3.6. Ітераційні цикли

Не у всіх задачах заздалегідь відома кількість повторень циклу. Прикладами таких задач можна вважати обчислення квадратного кореня із

цілих позитивних чисел, обчислення тригонометричних функцій за допомогою рядів, інтегралів наближеними методами, уведення набору даних змінної довжини. Також ітераційними методами можна розв'язувати багато задач, пов'язаних з плануванням діяльності в економіці та проектувальних задач.

В ітераційних циклах великого значення набуває визначення точності шуканого результату, що дозволяє встановити момент виходу із циклу, або врахуванням якої іншої умови.

3.3.7. Складні цикли

Як вже зазначалось, крім простих циклів, в обчислювальних алгоритмах застосовуються складні. Сморід, як правило, включають два й більше циклів, вкладених один в одного або розгалужених і що не перетинаються. Наприклад, такі циклі застосовуються у визначенні торб всіх елементів матриці.

Контрольні питання

1. Назвіть етапи підготовки до розв’язку задач на ПК, що виконуються без застосування комп'ютера.
2. Які основні властивості алгоритмів обчислювальних процесів?
3. Які процеси можуть описувати алгоритми?
4. З якою метою будують графічні схеми алгоритмів?
5. За якими правилами складають блок-схеми алгоритмів?
6. Які види блоків використовують у блок-схемах?

4. ОСНОВИ ІНТЕГРОВАНОГО СЕРЕДОВИЩА АЛГОРИТМІЧНОЇ МОВИ VISUAL BASIC

4.1. Загальні положення

Технологія роботи в середовищі **Visual Basic** має у своєї основі ідеї об'єктно-орієнтованого та візуального програмування. Ідея об'єктно-орієнтованого програмування полягає в інкапсуляції (об'єднанні) даних і засобів їх опрацювання (методів) у тип, який називається об'єктом. Прикладами об'єктів можуть бути елементи керування у діалоговому вікні: кнопки, списки, текстові поля тощо. Середовище візуального програмування Visual Basic – це графічна автоматизована оболонка навколо об'єктно-орієнтованої версії мови **Basic**.

4.2. Користувацька оболонка середовища розробки Visual Basic

Для завантаження середовища розробки Visual Basic (VB) та створення проекту потрібно зробити наступні операції:

користуючись головним меню операційної системи натиснути кнопку **Пуск→Программы→Microsoft Visual Basic→Visual Basic**, далі у діалоговому

вікні **New Project** вкладки **New** виділити **Standard.exe** та натиснути кнопку **Открыть.**

Після завантаження програми на екрані можуть з'явитись інструменти й вікна інтегрованого середовища розробки (див. рис. 4.1). Охарактеризуємо ці об'єкти.

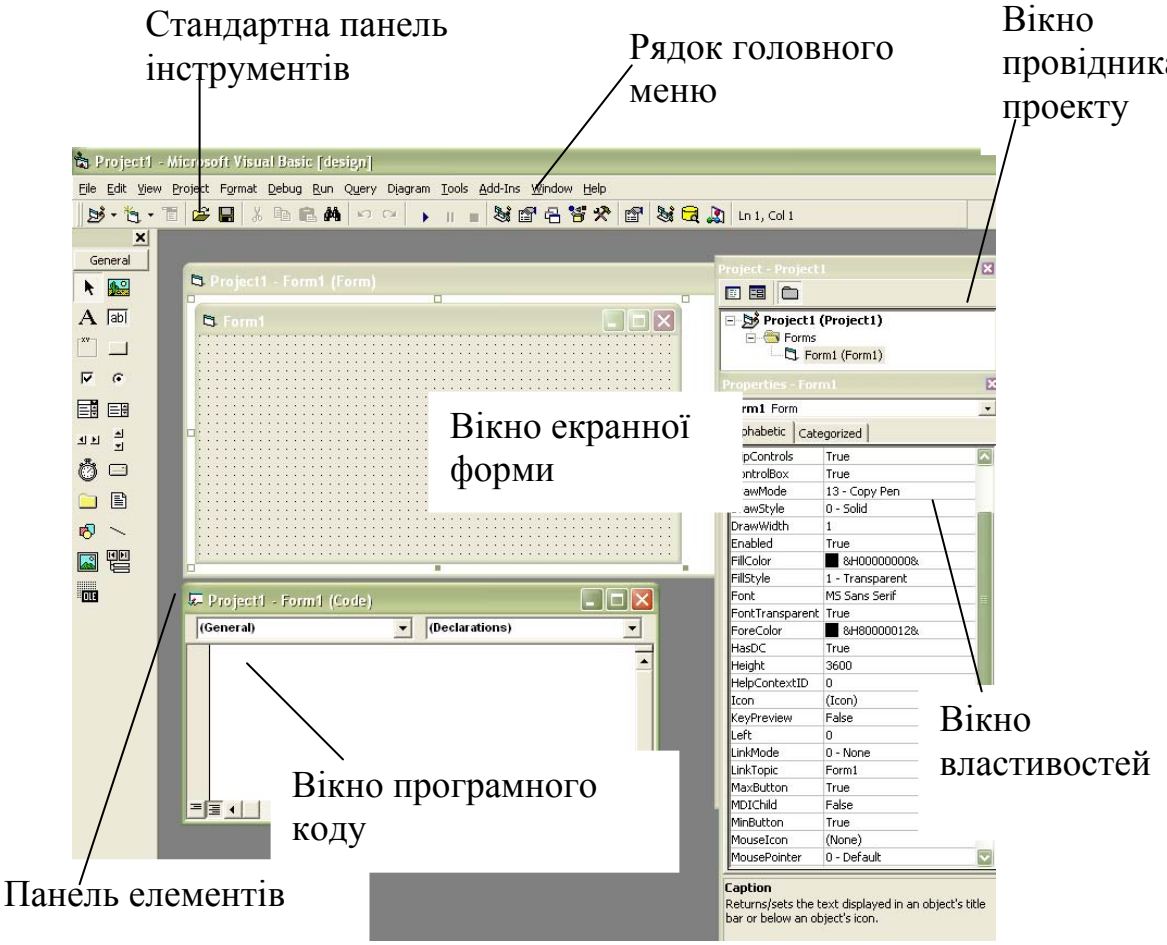


Рис. 4.1. Загальний вигляд вікна проектування в середовищі Visual Basic

1. Головне меню. Зазвичай користувацька оболонка має у своєму розпорядженні перелічені нижче засоби, що складаються з таких пунктів: **File, Edit, View, Project, Format, Debug, Run, Query, Diagram, Tools, Add-Ins, Window, Help**.

Зокрема меню File містить стандартні команди, призначені для роботи з файлами програмного середовища. За допомогою цих команд можна створити новий проект (**New Project**), відкрити чи закрити файл проекту (**Open Project**, **Remove Project**), додати проект (**Add Project**), зберегти проект чи форму (**Save Project, Save Project As, Save Form., Save Form. As**), роздрукувати вибрані файли проекту (**Print**) чи створити автономну виконувану прикладну програму exe-модуль (**Make Project.exe**).

Меню **Edit** включає стандартні команди для пошуку й заміни фрагмента тексту (**Find**, **Replace**, **Find Next**), команди для роботи з буфером (**Copy**, **Cut**, **Paste** тощо).

У меню **View** містяться команди для візуалізації елементів середовища.

Меню **Project** зосереджує команди, що використовуються в керуванні проектом, зокрема при додаванні файлів до проекту (**Add Form**, **Add Module** тощо).

За допомогою команд меню **Format** можна вирівнювати компоненти відносно сітки та між собою (**Align**), задавати порядок відображення компонентів, які перетинаються (**Order→Bring to Front**, **Order→Send to Back**), змінювати розміри вибраного компонента (**Make Same Size**) і т. ін.

Меню **Debug** призначається для налагодження проекту та пошуку помилок в програмному коді.

Меню **Run** включає команди, які керують роботою програми (**Run**, **Break**, **End** тощо).

У меню **Tools** розміщено команди для задання параметрів середовища VB.

Панелі інструментів служать для розташування кнопок деяких вище зазначених команд, набір команд регулюється потребами користувача.

2. Стандартна панель інструментів є основним засобом середовища Visual Basic. За її допомогою можна виконати широкий спектр операцій, передбачених у пунктах меню **File** (файл), **Project** (проект), **Debug** (налагодження) і **Run** (запуск). Призначення кожної з команд відображене в додатку 1.

3. Панель Form Editor призначається для переміщення елементів керування у формі, зміни їх розмірів, вирівнювання щодо меж форми. Перелік усіх елементів цієї панелі подано в додатку 2.

4. Вікно проектування **Visual Basic [design],** включає початковий рядок головного меню вікна й звичайні для нього кнопки, наприклад, кнопки згорнути в значок, розгорнути/відновити, закрити; та інші.

5. Вікно конструктора форм **Form** ***– екранна форма*** призначається для створення власного прикладного вікна користувача. У ньому обов’язково подаються характерні для всіх вікон ОС Windows назви вікна й відповідної кнопки системного меню.

6. ***Вікно інструментів пакета*** **Visual Basic,** що складається із двох стовпчиків кнопок інструментів, призначення яких відображене в додатку 3.

7. Вікно провідника проекту із заголовком **Project1,** у якому перелічуються всі компоненти створюваного додатка – екранні форми й програмні модулі.

8. ***Вікно властивостей об'єктів*** **Properties**, особливості якого розглянуто далі.

9. ***Вікно програмного коду*** **Code** призначено для створення в ньому програмних модулів (тобто самих програм, що виконують розрахунки, перетворення та певні дії).

Усі вікна мають кнопку системного меню й можуть бути закриті. Якщо на екрані немає потрібного вікна, його можна відкрити за допомогою таких команд пункту меню **View**:

- **Object** – вікно екрана "**Forml**";
- **Toolbox** – вікно інструментів пакета VB;

● **Properties Window** – вікно властивостей об'єктів "**Properties**";
● **Code** – вікно програмного коду "**Code**";
● **Project Explorer** – вікно провідника проекту.

Пакет містіть досить повну довідкову систему. Виклик її здійснюється через клавішу [**F1**] або пункт меню **Help.**

Крім перелічених вище, у вікні проектування можна викликати:

● вікно редактора меню **(Menu Editor)**, яке забезпечує створення й редагування рядка меню для форми;

● вікно макета форми (**Form Layout**) – для перегляду форми на екрані;

● вікно перегляду об'єктів (**Object Browser**) – для ознайомлення з усіма елементами проекту;

● вікно **Локальні (Locals)** – для перегляду значень локальних змінних величин;

● вікно **Спостереження (Watches);**

● вікно **Безпосереднє виконання (Immediate)** – призначене для ручного введення й виконання команд.

Вихід із середовища розробки **VB** можна виконати за допомогою системного меню вікна проектування (**Microsoft Visual Basic [design**]) або таким чином: пункт меню **File**→команда **Exit.**

4.3. Основні принципи об' єктно - орієнтованого програмування у середовищі Visual Basic

4.3.1. Загальні положення

Як уже зазначалось вище, алгоритмічна мова **Visual Basic** (**VB**) – це мова об'єктно-орієнтованого програмування, в якому можна маніпулювати готовими об'єктами й методами їхньої обробки на рівні операторів алгоритмічної мови.

В об'єктно-орієнтованій мові використовуються такі поняття як:

● **об'єкти;**
● **властивості;**
● **події;**
● **методи.**

4.3.2. Характеристика об'єктів середовища VB

Під об'єктами в середовищі **Visual Basic** розуміють пристрої або загальні елементи Windows-додатків, використовувані більшістю створюваних програм. При об'єктно-орієнтованому програмуванні практично всі компоненти комп'ютерної системи вважаються об'єктами. Зокрема, в середовищі Visial Basic виділяються такі групи об'єктів:

● ***глобальні (global objects)***, серед яких **clipboard** (буфер обміну), **debug** (налагодник), **printer** (принтер), **screen** (екран), **app** (додатки);

● ***інтерфейсні, або об'єкти взаємодії,*** наприклад, **form** (екранна форма), **controls** (керуючі елементи, зокрема ті, що присутні на панелі інструментів);

● ***об'єкти бази даних.***

У першу чергу будуть використовуватися об'єкти вікна панелі елементів керування (додаток 3).

Щоб ідентифікувати будь-який значок елемента на панелі інструментів **Visual Basic**, можна встановити в його місці покажчик миші, при цьому з'явиться спливаюча підказка із назвою цього елемента.

Складаючи власну програму кожен користувач має вирішити, з якими конкретно об'єктами він буде працювати, це стосується форми вікон, набору пристроїв, методу виведення результатів (на екран або на друк), а також які керуючі елементи повинні містити вікна програми.

Усі вимоги до обраних об'єктів користувач фіксує як **властивості** у вікні **Properties.** Кожному об'єктові відповідає свій заздалегідь заданий набір властивостей.

4.3.3. Властивість об'єктів

Властивість об'єкта – це якість або характеристика об’єкта що визначає його динамічні характеристики та зовнішній вигляд. Наприклад, властивостями об'єктів вважаються їхні імена, внутрішній вміст, колір фону й символ. Як правило, список властивостей кожного об'єкта складено заздалегідь, а користувач може встановлювати їхні конкретні значення, наприклад, колір – **синій**, тип шрифту – **Times New Roman**, зміст текстового напису й т. д.

Для встановлення значення властивості конкретного об'єкта необхідно клацнути лівою клавішею миші в місці цього об'єкта, тобто зробити його активним, і вивести клацанням на передній план вікно **Properties** (рис. 4.2). У вікні відображається список властивостей активного в даний момент об'єкта, а також уточнюються присвоєні за умовчанням цим властивостям значення.

Під рядком заголовка вікна розташований інший рядок (поле) із випадним списком, який містить перелік усіх об'єктів створюваного проекту. У цей рядок має бути внесено ім’я об’єкта, який нас цікавить.

Виклик необхідної властивості об’єкта для її встановлення здійснюється за його назвою *в першому стовпчику* вікна **Properties.** Щоб змінити значення властивості, потрібно клацнути лівою клавішею миші в місці *другого стовпчика* вікна, зокрема по правій кнопці наприкінці рядка цієї властивості у списку із припустимими значеннями, який при цьому з'являється. Далі за допомогою клацання вибирають необхідну властивість або активізують рядок і вводять дані за допомогою клавіатури.

Наведемо приклади деяких властивостей об’єкта:

● **Name** – встановлює ідентифікатор (ім'я) для доступу до об'єкта (текстового поля, поля написів, поля меню, які керують кнопкам і т. д.) із програмного коду. Ця властивість недоступна під час виконання програми. Наприклад, властивість Name – для форми встановлює її ім'я, використовуване в програмі. Ім'я починається з букви, має певний набір букв, цифр і підкреслення, загальна кількість яких не перевищує 40 символів. При цьому спочатку задається ім'я елемента, а потім для нього уводиться код програми;

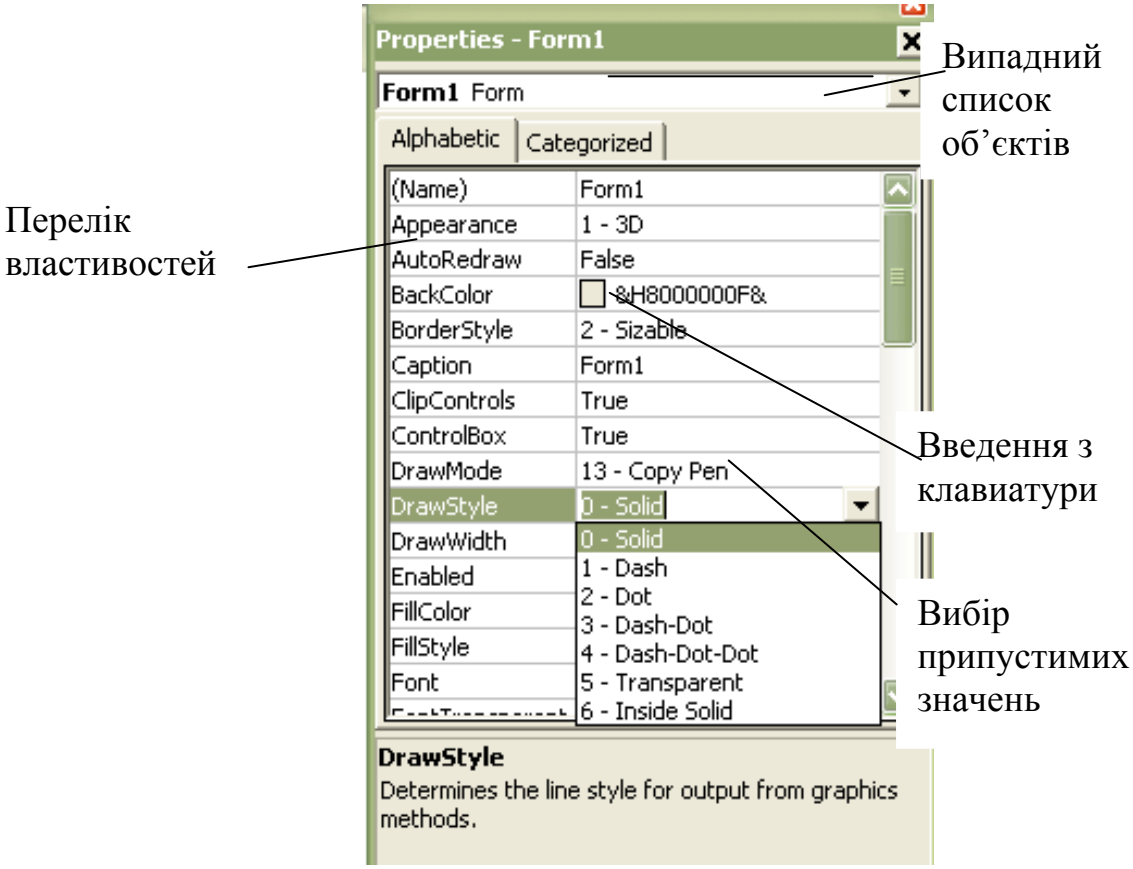


Рис. 4.2. Загальний вигляд вікна властивостей **Properties**

● **Alignment** – виконує вирівнювання тексту **(0** – вліво, **1** – управо, **2** – центрування).

● **Autosize** – автоматичне керування розміром поля напису або текстового поля **(true** – означає автоматичну зміну поля відповідно до розміру вмісту, **false** – розміри незмінні, а зайвий уміст відтинається).

● **BackColor, ForeColor** – установлення кольору фону й переднього плану об'єкта.

● **BorderStyle** – задає тип рамки для об'єкта, причому для ***form*** і ***textbox*** використовуються тільки для читання під час виконання програми. Можливі такі значення цієї властивості:

• **0** – немає контуру;

• **1** – фіксований одинарний верхній контур (рядка назви й меню);

• **2** – за умовчуванням змінюваний контур;

• **3** – фіксований подвійний.

● **Caption** – текст, відображуваний у заголовку (для форми), у середині або поруч із елементом керування.

● **FontName** – встановлює або повертає шрифт, який використовується для відображення тексту в елементах керування, а також при виконанні операцій малювання й друку. Рекомендується використовувати шрифт **Times New Roman Cyr**. Параметри **Height, Width** означають зовнішню висоту й ширину об'єкта. Задаються у твіпах (1 см = 567 твіпів).

● **Text** – забезпечує відображення інформації, що має вигляд пов’язаних між собою граматично і за змістом речень у текстовому або комбінованому полі або у полі списку. В останньому випадку передбачене тільки читання під час виконання програми.

4.3.4. Характеристика подій об’єктів

Visual Basic вважається мовою, яка орієнтується на обробку подій. Це означає, що окремі частини програми починають виконуватись у відповідь на певні ***події.*** Отже кожен об'єкт повинен реагувати на ці події (дії) такими засобами:

● на екрані (шляхом клацання миші у місці керуючих елементів);

● на клавіатурі (натисканням різних клавіш);

● у програмі автоматично під впливом конкретних даних.

Тобто програма повинна мати керовану подіями архітектуру.

Наведемо приклад типових подій:

● **Change** – відбувається при зміні стану елемента керування за умови настання цілого ряду можливих подій (так званого масиву подій);

● **Click** – відбувається при одноразовому клацанні клавішею миші на об'єкті;

● **DblClick –** відбувається при подвійному клацанні клавішею миші на об'єкті;

● **KeyPress** – відбувається при натисканні клавіші на клавіатурі;

● **Load** – відбувається під час виклику додатка за допомогою оператора **Load** у програмі або внаслідок неявного завантаження;

● **Unload** – відбувається внаслідок вивантаження форми з оперативної пам'яті через маніпуляції користувача (за допомогою меню або кнопок) або з програми за допомогою оператора **Unload**.

Настання кожної події в Visual Basic пов’язується з виконанням відповідної ***процедури (підпрограми)***. У загальному вигляді маємо такий синтаксис типової процедури:

```
Sub <ім’я об'єкта>_<ім'я події> (оголошення параметрів)
      Оператори
End Sub
```

Для кожного об'єкта зафіксовано перелік можливих процедур, завдання користувача – вибрати потрібну і записати відповідний їй програмний код, використовуючи спеціальне діалогове вікно (див. рис. 4.3).

4.3.5. Застосування методів у роботі з об’єктами

При роботі з розглянутими вище об'єктами виникає потреба у виконанні багатьох стандартних операцій, в яких задіяні відповідні вікна, кнопки, поля. **Visual Basic** звільняє користувача від програмування таких операцій шляхом надання можливості застосувати цілий ***набір методів.***

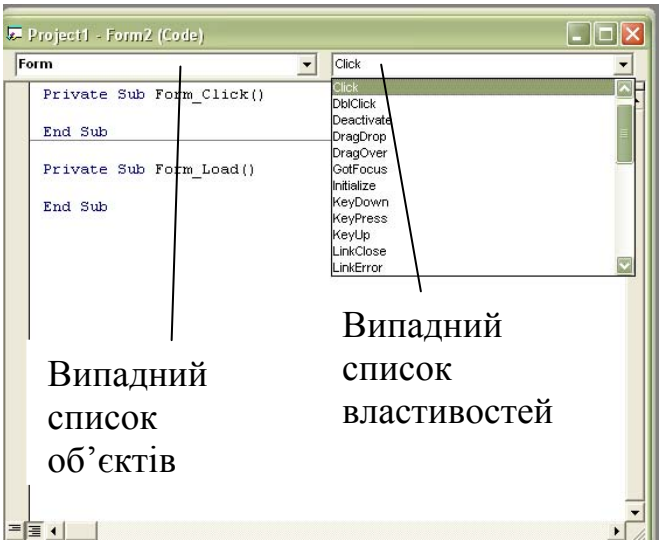


Рис. 4.3. Загальний вигляд вікна програмного коду

Наприклад, можна використовувати такі з них:

- **Clear** – очищає вміст буфера обміну (**Clipboard**);
- **Cls** – очищає форму або поле ілюстрацій;
- **EndDoc** – припиняє відправлення документа на принтер;
- **GetData** – повертає зображення з буферу обміну **Clipboard**;
- **GetText** – повертає текстовий рядок із буфера обміну;
- **Hide** – приховує форму, але не вивантажує її з оперативної пам'яті;
- **Move** – переміщає форму або об'єкт;
- **Print** – друкує текстовий рядок на об'єкті;
- **SetData** – поміщає ілюстрацію заданого формату у буфер обміну **Clipboard**;
- **SetText** – поміщає текстовий рядок у буфер обміну;
- **Show** – відображає форму на екрані.

Синтаксис запису методу має такий вигляд:

<ім’я об’єкта> <ім’я методу> [параметри]

Приклади:

- **Forml .Hide**;
- **Forml .Show**;
- **Printer.Print *"y = ";y.Text.***

Об'єкти, що мають спільні властивості й поведінку (до них відносять події і методи), поєднуються в ***класи.*** Ім'я класу, як правило, відображує тип об'єктів, які в нього входять. Класи можна розглядати як основу для створення інших об'єктів цього самого типу. Тому всі об'єкти одного класу, наприклад, **TextBox,** будуть діяти однаково. За умовчуванням об’єкти того самого класу мають ім'я, що складається з імені батьківського класу й порядкового номера, наприклад, **Text1, Text2, Text3.** При необхідності користувач може створювати свої класи об'єктів.

У **Visual Basic** також широко використовується поняття колекції об'єктів. ***Колекція*** – це набір об'єктів, об'єднаних загальним ім'ям, причому не

обов'язково вони можуть входити в один клас. Для прикладу згадаємо шість вбудованих в Visual Basic 6 колекцій, у т. ч. **Forms** і **Controls.** Зокрема **Forms** містить безліч завантажених форм додатка, **Controls** – множину всіх елементів керування у формі. Користувач може створювати власні колекції, поповнювати й актуалізувати їх.

Розробка нової програми або, за термінологією Visual Basic, проекту ***складається із двох етапів:***

● ***Етап візуального програмування*** – передбачає ***створення форми або набору форм,*** що будуть видаватися користувачеві під час роботи з програмою. Погодження переліку й змісту форм із замовником може значно зекономити кошти, витрачені на розробку проекту.

● ***Етап програмування у вхідному коді*** полягає у ***створенні власної програми обробки інформації,*** яка буде відповідати різним діям користувача.

4.4. Створення форм і встановлення властивостей

На етапі візуального програмування необхідно:

● визначити скільки екранних форм буде в задачі, з якою метою вони будуть використовуватися, які об’єкти них буде розташовано (текстові поля, кнопки, лінійки прокручування, прапорці і т. д.);

● створити ці форми;

● визначити властивості об'єктів (самих форм і розташованих на них елементів керування), зокрема це стосується зовнішнього вигляду й особливостей їхнього використання в програмі.

Потрібні елементи діалогового вікна зображуються на екранній формі за допомогою відповідних кнопок панелі інструментів. Для цього потрібно клацнути лівою клавішею миші на місці кнопки з необхідним інструментом, далі, не відпускаючи клавішу визначити розмір і місце розташування елемента у формі. Розміри самого вікна і розташованих на ньому елементів, коли вони активні (виділені), у процесі появи зображення об’єктів. Щоб зробити об'єкт активним (виділити), необхідно клацнути по ньому клавішею миші. При цьому навколо цього об’єкта з'явиться рамка виділення, що містить вісім маленьких квадратиків. Розміщуючи курсор миші всередині рамки виділення, можна переміщати об'єкт, а натискаючи клавішу миші на одному із квадратиків рамки – змінювати його розміри.

Для видалення непотрібного об'єкта спочатку необхідно зробити його активним, а потім натиснути клавішу **Delete** на клавіатурі.

Приклад 4.1

Завдання: Автомобіль починає рухатись із прискоренням. Шлях, пройдений автомобілем у метрах, розраховується за такою формулою:

$\mathbf{S = 1{,}1 \cdot t^2}$,

де **t** – час руху автомобіля, с.

Швидкість автомобіля (км/год) протягом кожної секунди від початку руху визначається за таким виразом:

V = 7,92·t.

Спроектувати форму для розрахунку відстані, пройденої автомобілем, і його швидкості в заданий момент часу.

Виконання. На екранній формі (рис. 4.4) повинні розташовуватися текстове вікно для введення значень величини **t**, а також вікно для виведення результатів обчислення величин **S** та **V**. Щоб знати місце розташування вікон, їх необхідно підписати, створивши до них поля написів. Розрахунок має виконуватися шляхом клацання лівої клавіші миші по кнопці **Розрахунок,** а завершується операція клацанням у місці по кнопки **Вихід**. Використовуючи кнопку "**Очищення текстових вікон**", звільняють їх від зайвої інформації.

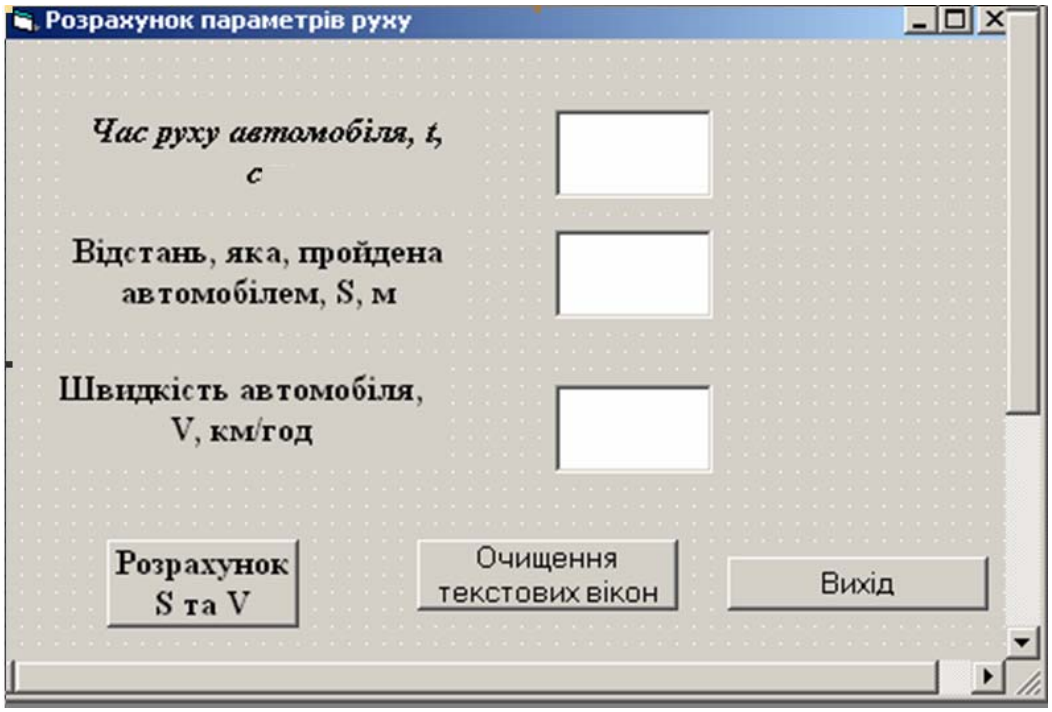


Рис. 4.4. Вікно форми проекту – "Розрахунок параметрів руху"

Виконання цього прикладу забезпечується переліченими нижче операціями.

1. Намалюйте на папері або уявіть собі подумки вигляд такого вікна, після цього можна розпочати до створення екранної форми на комп'ютері.

2. Проведіть завантаження середовища Visual Basic.

3. У відкритому діалоговому вікні **Visual Basic** з'явиться порожня екранна форма **Form1.** Якщо її розміри не задовольняють користувача, то варто клацнути лівою клавішею миші по ній (виділити) і змінити розмір.

4. Далі розробляємо форму програми, для чого необхідно:

● *дати ім'я електронній формі,* при цьому:

● відкрити вікно **Properties**, а якщо воно відсутнє на екрані, то скористатись командою пункту меню **View → Properties Windows**;

• вибрати властивість **Caption**, у правому стовпчику стерти **Form1** і написати **Розрахунок параметрів руху;**

● *записати вихідні данні й результат* розрахунків таким чином:

• на панелі інструментів виконати подвійне клацання лівою клавішею миші по кнопці  **Label** – напис, у центрі форми з'явиться прямокутник з написом **Label1**, перетягнути його на спеціальне відведене у створюваній формі місце (рис. 4.4), далі двічі повторити цю послідовність дій, одержавши написи **Label2**, **Label3**;

• активізувати клацанням миші вікно **Properties**; де у верхньому списку, який при цьому з'явився, вибрати напис **Label1**; а в списку властивостей виділити **Caption**, при цьому в другому стовпчику стерти **Label1** і написати: **Час руху автомобіля, t**;

• у верхньому списку випадного вікна **Properties**, вибрати напис **Label2**, виділити властивість **Caption**, написавши: **Відстань, пройдена автомобілем, S**;

• у верхньому списку випадного вікна **Properties**, вибрати напис **Label3**, виділити властивість **Caption**, написавши: **Швидкість автомобіля, V**.

● *створити текстові вікна для введення і виведення даних* шляхом запровадження таких дій:

• на панелі інструментів виконати подвійне клацання лівою клавішею миші по кнопці  **TextBox** для створення текстового вікна **Text1**, далі потрібно перетягнути його на спеціально відведене у створюваній формі місце (рис. 4.4), потім двічі повторити цю послідовність дій, одержавши написи **Text2** і **Text3**;

• активізувати клацанням миші вікно **Properties**; де у верхньому списку, який при цьому з'являється, вибрати напис **Text1**; а в списку властивостей виділити **Name**, стерти **Text1** і написати **t;**

• потім у списку властивостей вибрати властивість **Text** і стерти в іншому стовпчику напис **Text1** (поле має бути вільним від написів), далі аналогічно присвоїти елементові **Text2** ім'я ***S***, а **Text3** – ***Y*** і, виділивши властивість **Text**, стерти написи **Text2** і **Text3**.

● *створити екранні кнопки* таким чином:

• на панелі інструментів виконати подвійне клацання лівою клавішею миші по кнопці  **(CommandButton** – командна кнопка) і перетягнути її на спеціальне відведене у формі місце, далі повторити описані дії, переміщуючи дві інші кнопки;

• активізувати клацанням миші вікно **Properties**; де у верхньому списку, який при цьому з'являється, вибрати елемент **CommandButton1**; далі у списку властивостей виділити **Caption**, стерти **CommandButton1** і написати **Розрахунок;** потім виділити **Name** і замість **CommandButton1**, написати **Rozraxunok**;

• у верхньому випадному списку вікна **Properties**, вибрати елемент **CommandButton2**; далі, виділивши властивість **Caption** написати **Очищення текстових вікон**, а потім виділити **Name** зробити напис **Ochistka;**

• у верхньому випадному списку вікна **Properties**, вибрати елемент **CommandButton3**; потім виділивши властивість **Caption**, написати **Вихід**, а виділивши **Name** зробити напис **Vuxid.**

5. Зберегти створену внаслідок виконаних дій форму клацнувши лівою клавішею миші по піктограмі  стандартної панелі інструментів.

6. Двічі клацнути мишею у місці кнопки **Розрахунок**, що зумовлює появу вікна програмного коду **(Code),** у якому:

● лівий випадний список (**Object**) – містить перелік усіх елементів проекту;

● правий випадний список (**Procedure**) – назви процедур, які відповідають цим елементам.

7. Отже, дали вибираємо:

● у лівому списку елемент **Rozraxunok,** а в правому – процедуру **Click**, при цьому буде відображено програмний код процедуру з операторами початку і кінця процедури згідно до синтаксису **VB** (рис. 4.5), оскільки вхідні дані й результати вводяться через текстові вікна, а в програмі розрахунок відбувається із числами, то використовується функція **Val** для перетворення текстових даних у числові;

```
'розрахунок  відстані, яка пройдена автомобілем
'та його швидкості
Private Sub Rozraxunok_Click()
  Text2.Text = 1.1 * (Val(Text1.Text)) ^ 2
  Text3.Text = 7.92 * (Val(Text1.Text))
End Sub
'очищення текстових вікон
Private Sub ochistka_Click()
  Text1.Text = ""
  Text2.Text = ""
  Text3.Text = ""
End Sub
'вихід з програми
Private Sub vuxid_Click()
  End
End Sub
```

Рис. 4.5. Вікно програмного коду проекту – "Розрахунок параметрів руху"

● у лівому випадному списку, елемент **ochistka,** а в правому – **Click**, та вводимо оператор закінчення роботи програми **End;**

● у лівому випадному списку, елемент **Vuxid,** а в правому – **Click**, та вводимо написи:

Text1.Text = ""
Text2.Text = ""
Text3.Text = ""

8. Виконуємо написану програму в такій послідовності:

меню **Run** → команда **Start** → ввести у вікно **Час руху автомобіля, t, с,** – число **10,** клацнути мишею по кнопці **Розрахунок S та V** у вікні **Відстань, яка пройдена автомобілем, S, м** повинно вийти число **110,** а у вікні **Швидкість автомобіля, V, км/год.** – число **79,2.** Якщо в розрахунку допущено помилку, то про це видається повідомлення, а в тексті програми помилковий оператор виділяється кольором. Варто виправити помилку й знову спробувати виконати програму, уведенням меню **Run** → команди **Continue**.

9. Робота з програмою завершується клацанням лівою клавішею миші по кнопці **Вихід.**

4.5. Програмування процедур, пов'язаних з подіями

4.5.1. Загальні положення

При роботі користувача з формою після уведення вхідних даних клацання мишкою по кнопці **Розрахунок** повинне забезпечити одержання результату. Така реакція на подію "**клацання**" (**Click**) забезпечується на другому етапі проектування – ***програмування процедур, пов'язаних з подіями.***

Програмування процедури такого типу виконується у вікні програми **(Code Window),** що викликається подвійним клацанням лівою клавіші миші у місці потрібного об'єкта. При цьому використовуються як методи, так і звичні для алгоритмічних мов команди. Тому коротко розглянемо основні елементи мови **Visual Basic**.

Алфавіт мови складається з латинських букв, цифр, службових символів.

Наприклад, ***одинарні лапки*** позначають ***початок коментаря,*** який при виконанні програми ігнорується й несе тільки довідкову інформацію для її читання; знак підкреслення наприкінці рядка оператора свідчить про наявність його продовження на наступному рядку.

У програмах використовуються такі засоби:

- ***оператор присвоювання;***
- ***арифметичні операції в*** порядку пріоритетів їхнього виконання, наприклад: $\mathbf{a^x}$ – піднесення до ступеня, ***** – множення **,** / – ділення, \ - ділення з ігноруванням дробової частини результату, + – додавання, — – віднімання;
- ***операції відношень*** **– < , > , = , >=, <=, <>;**
- ***логічні операції*** **– And, Or, Not, Xor** та ін.;
- ***строкові вирази*** **–** текст, заведений у подвійні лапки.

4.5.2. Характеристика типів даних VB

Мова програмування Visual Basic використовує різні дані, типи яких охарактеризовано в таблиці 4.1.

Усі дані, що використовує програма, можуть бути ***сталими (константами)*** або ***змінними.*** Константи й змінні оголошуються на початку будь-якої підпрограми або модуля таким чином:

- ***константи*** – за допомогою оператора **Const;**
- ***змінні*** – за допомогою оператора **Dim.**

Оператор **Const** має такий синтаксис:

[Global] Const <ім'я> = вираз [,<ім'я> = вираз]...

Наприклад: **const Pi = 3.1415**

const Max% = 250

const Clovo$ = "відмінно"

Типи констант і виразів, що їм присвоюються, повинні збігатися. В арифметичних виразах заборонено використовувати операцію піднесення в степінь і деякі функції.

Таблиця 4.1

Типи даних мови програмування Visual Basic

Назва	Познач.	Характеристика типу даних
1. **Integer**	%	Цілі числа від –32768 до +32768, 2 байти
2. **Single precision**	!	Число з плаваючою крапкою одиночної точності, 4 байти (використовується за умовчуванням)
3. **Long Integer**	&	Довге ціле число, 4 байти
4. **Double precision**	#	Число з плаваючою крапкою подвійної точності, 8 байтів
5. **String**	$	Рядок до 65535 символів або оголошеної довжини
6. **Cuurency**	@	Число з фіксованою крапкою (15 знаків ліворуч і 4 праворуч від крапки, 8 байтів)
7. **Variant**		Дані, тип яких не відомий, займають більше місця, ніж явно оголошені змінні.

Оператора **Dim** має такий синтаксис:

Dim <ім'я змінної>[([опйс масиву змінних])] [**As [New] type**]

[, **<ім'я змінної>[([опис масиву])]**[**As[New]type]...**

При цьому

As type – визначає тип даних (**Integer**, **Long**,...) чи об'єкта. Наприклад:

Dim a As Integer,

Dim Temper As Single,

Dim d, b, rakurc – за умовчуванням тип **Variant**,

Dim FIO As String – будь-яка довжина символьної змінної,

Dim Street As String*75 – явна вказівка на довжину строкового типу (75 – кількість символів у назві вулиць).

New – використовується для створення нових характеристик спеціальних типів об'єктів, таких як **Form1**.

Опис масиву змінних включає зазначення їхньої нижньої та верхньої меж, а саме:

Dim A(8,3);

Dim A(0 To 8, 0 To 3)

Dim A(8,0 To 3)

Dim A(–4 To 10)

Максимальне число масивів у даному операторі – 60. Секція оголошень має бути на початку кожної процедури й процедури – функції. Оголошення змінних і масивів у процедурі, пов’язаною з подією, поширюється тільки на неї.

У записах формату операторів використовують такі позначення:

{} – вибір одного з перерахованих значень;

[ ] – елемент може бути або не бути;

<> – треба написати конкретне значення.

Для оголошення ***глобальних*** змінних використовують глобальний модуль **(General),** що автоматично включається до складу будь-якого проекту і розміщується перед першою процедурою.

Наприклад:

General
Option Explicit
Dim X As Integer
Dim I,Y

Якщо тип даних для змінної не визначено, то за умовчуванням її відносять до типи **Variant.** Це значить, що тип змінної визначається залежно від типу даних, які вперше будуть у неї поміщені.

4.5.3. Уведення – виведення даних

Уведення даних здійснюється через керуючі елементи форми – текстові поля, табличні поля (масиви), комбіновані списки (файли), зокрема шляхом набору на клавіатурі комп'ютера потрібних символів, а закінчується натисканням клавіші **Enter**. Також дані можна вводити клацаючи лівою клавішею миші у місці потрібного імені файлу, необхідних характеристик, керуючої кнопки **ОК** і т. д. При цьому переліченим подіям можуть відповідати підпрограми (процедури), за допомогою яких відбувається розв'язування задач.

Для введення даних у діалоговому режимі використовується функція **InputBox**, для якої характерний такий синтаксис:

InputBox ("Рядковий вираз" ["Заголовок"]["За умовчуванням"]).

Рядковий вираз – найчастіше являє собою будь-які текстові данні, найчастіше підказку, обов'язковий аргумент; **Заголовок** – заголовок вікна **InputBox**, необов'язковий аргумент; **За умовчуванням** – це текстовий рядок, що вводиться за умовчуванням, і являє собою необов'язковий аргумент**.**

Приклад використання функції **InputBox** (рис. 4.6):

User = InputBox ("Введіть ім'я файлу", "Створити файл", "New_File")

Виведення даних можна здійснювати на екран, на друк, у файл. У першому випадку це відбувається, наприклад, шляхом присвоєння певних значень текстовим, табличним полям.

Для виведення даних на поверхню об'єкта (поверхню форми, об'єкту **PictureBox**) необхідно використовувати метод **Print**, для якого характерне такий синтаксис:

[Об'єкт.] Print вираз

при цьому **об'єкт** – в даному випадку означає принтер (Printer), поверхню форми, об'єкт **PictureBox**; **вираз** являє собою матеріал, який видається на друк або на екран монітора.

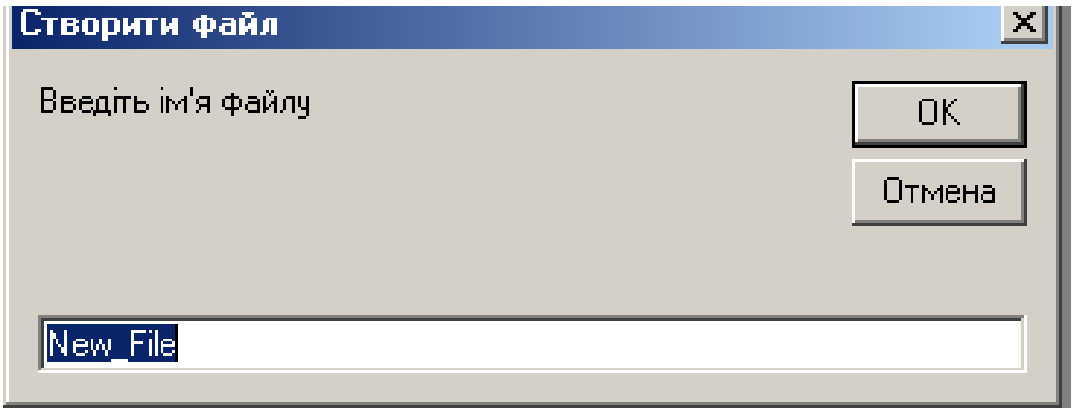


Рис. 4.6. Загальний вигляд вікна функції **InputBox**

Щоб виведені данні набули певної форми, необхідно використовувати так зване форматоване виведення – функцію **Format.**

Синтаксис цієї функції такий:

Format[$] (expression [,frm]),

де **$** – ознака рядкового типу, **expression** – арифметичний або рядковий вираз, **frm** – задання формату виведеного виразу, що відбувається за допомогою таких символів:

- **0** – друк цифри або нуля, наприклад, 000000, тобто буде надруковано число яке має не більше шести знаків, якщо знаків менше, то друкуються головні нулі.
- **#** – друк цифри або пробілу;
- **.** – друк місця десяткової крапки, наприклад, ####.##;
- % – виводить число в процентному вигляді;
- , – множення виразу на 1000;
- **E** – науковий формат у вигляді числа із плаваючою комою;
- **$** – рядковий формат " ";
- **:** – формат часу;
- / – формат дати та ін.

Наприклад, якщо Y = 12,454457, то з такого виразу:

Form1.Print "Y="; (Format(Y, "0.00")), на поверхню форми буде виведено, що **Y = 12,45**.

При введенні й виведенні даних може виникнути необхідність у зміні їхньої форми. Наприклад, введення й виведення у даних у текстове поле відбувається за рядковим типом, а розрахунки належить виконувати, використовуючи числові дані.

Перетворення строкових даних у числові здійснюється за допомогою функції **Val**, для якої характерне такий синтаксис:

Val (рядковий вираз) – числове значення.

Наприклад:

У текстове поле **a** введено рядок 5 (символ 5 сприймається як текст). Для виконання розрахунків уміст поля **a** необхідно перетворити в число, тобто

Val(a.Text) = 5.

Перетворення числових даних у рядкові здійснюється за допомогою функції **Str.**

Синтаксис цієї функції такий:
Str (числове значення) → рядковий вираз
Наприклад: **y.Text = Str(S) + "грн"**.

4.5.4. Надання привабливості формі та засоби створення виконавчого файлу

Створюючи проекти у середовищі Visual Basic студент повинен подбати про зовнішню привабливість електронних форм, які видаються в процесі розв'язання задач. Навіть найпростіші форми (скажімо так, як на рисунку 4.5), можна подати в більш естетичному вигляді, зокрема використати для зображення фону й елементів різні кольори, зробити вказівку на формулу, яка застосовується в розрахунку, вставити картинку (графічний об'єкт). Вирівнювати розміри або вміст полів, зафарбувати об'єкти можна групою, використовуючи групове їхнє виділення за допомогою клавіші Ctrl і клацання клавішею миші.

Наприклад, для ***зміни кольору фону форми або кожного з її елементів*** необхідно:

- виділити клацанням миші потрібний об'єкт;
- у вікні **Properties** знайти властивість **BackColor** і з переліку зразків у другому відповідному списку **(вкладка** Palette) підібрати потрібний колір.

Усі написи й отримані результати можна ***центрувати*** у межах відповідних об'єктів. Для цього присвоюють властивості **Augment** кожного об'єкта властивість **2** – **center** взяту з другого випадного списку. Для того щоб ***уставити*** попередньо обрану чи намальовану ***картинку у вигляді фону,*** варто виділити форму, потім знайти й клацнути лівою клавішею миші у місці властивості **Picture** і за допомогою кнопки наприкінці рядка викликати діалогове вікно **Load Picture,** у якому позначити папку (наприклад, **ClipArt)** та файл, де розміщено картинку. В екранну форму малюнок уставляється з верхнього лівого кута. Малюнок не може пристосовуватись до форми тому, якщо він більшій від неї, то відобразиться не весь. Збільшуючи форму, збільшують видиму частину малюнка.

Щоб видалити малюнок із форми, необхідно в пункті **Забрати картинку** вікна властивостей очистити сторінку властивостей **Picture**.

Щоб ***уставити картинку в окремо відведене місце*** можна використовувати об'єкти **Picture** або **Image** (активізуються за допомогою відповідних кнопок панелі інструментів аналогічно до встановлення раніше розглянутих об'єктів). Об'єкт **Picture** дозволяє редагувати малюнок, а об'єкт **Image** – ні. Далі вибирають властивість **Picture** і виконують ту саму послідовність дій, що й для форми. Або можна скопіювати потрібний малюнок у буфер, а потім, клацнувши лівою кнопкою миші в місці об'єктів **Picture** або **Image,** виконати пункт меню **Edit** → команда **Paste.**

Для вставлення формул або об'єктів з інших додатків можна використовувати інструмент **OLE** (кнопка **OLE** на панелі інструментів). У діалоговому вікні **OLE** необхідно вибрати потрібну програму (об'єкт), наприклад, **Microsoft Equations.**

Після виконання всіх описаних дій файл треба зберегти, а всю виконану роботу ***роздрукувати***, запровадивши такі операції: виклик меню **File** → команда **Print** → вибір прапорців **Form Image** (для друку електронної форми), **Code** (друк коду програми), **Form As Text** (друк властивостей усіх об'єктів проекту). Можна друкувати всі елементи проекту, а можна вибирати для друку будь-яке потрібне поєднання об’єктів.

Як уже зазначалось, створений проект виконується тільки в середовищі Visual Basic. Щоб зробити його незалежним від цього середовища, варто створити виконавчий файл (із розширенням **exe). Exe-файл** може безпосередньо запускатися з середовища ОС Windows як додаток, шляхом подвійного клацання лівою клавішею миші у місці відповідного значка. Якщо користувач хоче присвоїти своїй програмі нестандартний значок файлу, то він повинен установити властивість **Icon** для всієї форми (кнопкою в другому стовпчику відкрити діалогове вікно й вказати на папку та файл із потрібним значком).

4.5.5. Використання лінійок прокручування

Для зручності роботи під час виконання навігаційних операцій за списком або при обчисленні значення змінної величини у програмах можна використовувати ***лінійки прокручування.*** Їх можна помістити у форму за допомогою таких елементів керування з панелі елементів:

 – горизонтальна лінійка **(HScrollBar)**;

 – вертикальна лінійка **(VScrollBar)**.

Переміщення бігунка на лінійці прокручування зумовлює зміну властивості **Value.** Ця властивість визначає поточну позицію для елемента (конкретне його значення), який перебуває в діапазоні, властивостей **Max** і **Min** (стосується цілих чисел).

З лінійками прокручування пов'язані ***події:***

Change – настає в момент клацання мишею по кнопках зі стрілками або між кнопкою зі стрілкою та бігунком, а також у момент відпускання бігунка після його переміщення;

Scroll – дозволяє одержати значення властивості **Value** при переміщенні бігунка до настання події **Change.**

Крок переміщення при клацанні мишею по кнопці зі стрілкою задається властивістю **SmallChange,** а при клацанні між кнопкою зі стрілкою та бігунком – властивістю **LargeChange.**

Контрольні питання

1. Дайте характеристику об'єктно-орієнтованих алгоритмічних мов на прикладі Visual Basic.
2. Що являє собою об'єкт у середовищі Visual Basic, які приклади різних типів об'єктів ви знаєте?
3. Якими ознаками характеризуються об'єкти в середовищі VB і яким чином вони задаються?
4. Чим відрізняються поняття “події” і “методу” у середовищі Visual Basic? Проілюструйте цю різницю.
5. Розкрийте суть поняття “клас” і як воно співвідноситься з поняттям “об'єкт” у VB?
6. Які етапи створення проекта в середовищі VB ви знаєте?
7. Що являє собою візуальне програмування?
8. Як можна визначити поняття програмний код у Visual Basic?

5. Оператори в середовищі Visual Basic

5.1. Оператор присвоювання

Перш ніж описувати формати операторів приведемо основні відомості про такі складові кожної програми, як оператори, операнди, операції.

Оператор - синтаксична конструкція мови програмування, призначена для визначення дій з певних правил, що реалізовуються при виконанні програм. Зазвичай оператори класифікуються на прості (наприклад оператор безумовного переходу **Goto**), включаючих прості дії, і складені (наприклад оператор циклу **For…Next**), включаючі як компоненти інші або такі ж оператори.

Операнд - елемент даного, який представляється вираженням і над яким виконуються операції.

Операція - обмежувач або зарезервоване слово, які використовуються для вказівки алгоритмів перетворення значень одного або двох операндів в результат деякого типу. Є шість класів операцій :

- логічні операції і форми управління з проміжною перевіркою;
- операції відношення і перевірки приналежності;
- бінарні адитивні операції;
- унарні адитивні операції;
- мультиплікативні операції і операції вищого пріоритету.

Більшість програм реалізується шляхом виконання операторів, які є інструментом, що керують ходом обчислювального процесу.

У мові Visual Basic найпростішим вважається ***оператор присвоювання.***

Він має такий формат:

<змінна> = вираз

Оператор присвоювання пов’язується знаком рівності з конструкцією, у якій значення виразу, що стоїть справа, присвоюється змінній, ім'я якої

написано зліва. Наприклад, у результаті виконання поданої нижче пари операторів змінній z буде присвоєно значення **a + b**, тобто

x = a

z = x + b.

Оператор присвоювання допускається використовувати також для маніпуляції об'єктів. Проте якщо змінній присвоюється значення посилання на об'єкт, то до оператора присвоювання слід додати ключове слово **Set**, наприклад:

Set objA = objB.

Тут передбачається, що **objA** і **objB** були раніше оголошені як змінні об'єктного типу, і змінна **objB** у даний момент містить посилання на деякий об'єкт. Унаслідок виконання наведеного вище оператора змінній **objA** присвоюється значення посилання на той самий об'єкт.

5.2. Арифметичні оператори

У мові Visual Basic арифметичні оператори дозволяють виконувати обчислювальні арифметичні операції у повній відповідності із правилами арифметики (див. табл. 5.1).

Таблиця. 5.1

Знаки арифметичних операцій, використовуваних у мові Visualasic

Знак операції	Опис
+	Додавання
–	Віднімання
*	Множення
/	Ділення
\	Ділення без залишку
^	Піднесення до степеня
mod	Залишок від ділення за модулем

Виконуючи арифметичні операції, потрібно враховувати такі умови:

- знак операції додавання можна використовувати при побудові арифметичних виразів з даними типу **Date**;
- якщо в операції додавання використовуються дані типу **Integer** і **Long**, то результат обчислення виразу матиме тип даних **Long**;
- якщо в операції віднімання один з операндів має тип даних **Date**, то результат обчислення виразу матиме тип даних **Date**;
- якщо обидва операнди в операції віднімання мають тип даних **Date**, то результат віднімання виразу матиме тип даних **Double**;
- якщо в операції множення використовуються дані різних типів, то результат обчислення виразу матиме тип даних, відповідний типу даних, що має найбільшу точність;

● при множенні значення змінних, що мають тип даних **Variant**, і містять одночасно значення типу **Date**, будуть перетворені в числові;

● якщо при виконанні ділення виявиться дільник, що дорівнює нулю, то буде видано повідомлення про помилку;

● для результату обчислення операції ділення чисел з плаваючою крапкою зазвичай використовується тип даних **Double**, а для чисел типу **Integer** – **Single**;

● результатом операції ділення без залишку буде ціле число без округлення, при цьому дробова частина результату відкидається;

● операція піднесення до степеня (^) припускає піднесення значення першого операнда виразу до степеня, який дорівнює значенню другого операнда;

● операція ділення за модулем передбачає ділення першого операнда на другий і, як результат, повертається цілочислова остача цього ділення.

5.3. Логічні оператори

Логічні оператори в мові Visual Basic використовуються для маніпулювання логічними значеннями – **True** (у числовому вираженні це 1) і **False** (у числовому вираженні це 0). У табл. 5.2 стисло описані логічні оператори, підтримувані мовою Visual Basic.

Таблиця 5.2

Допустимі логічні оператори мови Visual Basic

Оператор	Опис
And	Кон'юнкція (логічне І)
Or	Диз'юнкція (логічне АБО)
Not	Заперечення (логічне НЕМАЄ)
Xor	Виняток (що логічне виключає АБО)
Eqv	Логічна еквівалентність
Imp	Логічна імплікація

Оператор **And** позначає операцію кон'юнкції, результатом виконання якої буде значення **True** тоді й тільки тоді, коли обидва оператори мають це значення. У решті значень результатом виконання цієї операції буде значення **False**. Наприклад, вираз **(3 > 2) And (4 > 3)** має значення **True**, а вираз **(3 > 6) And (4 > 3)** має значення **False**.

Логічний оператор **Or** позначає операцію диз'юнкції, результатом виконання якої буде значення **True**, якщо принаймні один з операндів має значення **True**. Результат матиме значення **False** тоді й тільки тоді, коли обидва операнди мають це значення. Таким чином:

(3 > 5) Or (4 > 2) – значення **True**.

(2 > 3) Or (2 > 4) – значення **False**.

Логічний оператор **Not** позначає операцію логічного заперечення і має тільки один операнд. Результатом виконання цієї операції буде значення **True**, якщо операнд має значення **False**, і навпаки, результатом буде значення **False**,

якщо операнд має значення **True**. Наприклад, вираз **Not (5 > 3)** має значення **False**, оскільки вираз в дужках має значення **True**.

Логічний оператор **Xor** позначає логічну операцію **що "виключає АБО"**, результатом виконання якої буде значення **True** (якщо операнди мають різні значення), і значення **False**, якщо обидва операнди мають значення **True**.

Логічний оператор **Eqv** позначає логічну операцію еквівалентності двох виразів, результатами виконання якої буде значення **True**, якщо обидва операнди мають значення **True** або обидва **False**, і значення **False**, якщо тільки один з операндів має значення **False.**

Логічний оператор **Imp** позначає логічну операцію імплікації (проходження) двох виразів, результатом виконання якої буде значення **False**, якщо перший операнд має значення **True**, а другий операнд має значення **False**, і значення **True** у всіх інших випадках.

5.4. Оператори порівняння

Оператори порівняння використовуються в мові Visual Basic для порівняння числових і рядкових значень змінних величин, констант і результатів обчислення виразу. Результатом виконання операції порівняння завжди буде значення типу **Boolean**: або **True** (істина), або **False** (неправда). У таблиці 5.3 описано оператори порівняння, що підтримуються мовою Visual Basic.

Таблиця 5.3

Оператори порівняння в мови Visual Basic

Оператор	Опис
=	Дорівнює
>	Більше
<	Менше
>=	Більше або дорівнює
<=	Менше або дорівнює
<>	Не дорівнює
Is	Порівняння двох операндів, що містять посилання на об'єкти
Like	Порівняння двох рядкових виразів

Якщо обидва операнди у виразі мають один і той самий тип даних, то в мові Visual Basic виконується їхнє просте порівняння. Якщо один або обидва операнди являють собою змінні величини типу **Variant**, то компілятор Visual Basic перетворить тип **Variant у** який-небудь сумісний тип даних, інакше на екран буде виведено повідомлення про помилку під час виконання програми.

Знаки операції порівняння можна використовувати як для чисел, так і для рядкових значень. Порівнюючи останні, компілятор Visual Basic послідовно зчитує окремі символи зліва направо, визначаючи старшинство в алфавітному порядку, а будь-яких інших символів відповідно до двійкового значення коду ASCII. При цьому буде визнано, що один рядок дорівнює іншому, якщо вони мають однакову довжину й містять одні й ті самі символи, розташовані в

однаковому порядку. Результатом такого порівняння буде значення **True**. У разі одна із перерахованих умов не буде виконана, то результатом порівняння рядків буде значення **False**.

Мова Visual Basic має у своєму розпорядженні два способи порівняння символів різних регістрів:

Порівняння рядків у двійковому режимі (метод порівняння за умовчуванням). Порівнюючи рядкові дані цим методом, використовують дійсний двійковий еквівалент коду кожного символу. Букви верхнього регістра мають менший двійковий код, ніж букви нижнього регістра. Наприклад, для букви "**A"** код символу верхнього регістра (**A**) менший від коду символу нижнього регістра (**a**) тобто" **a" > "A" True.**

Порівняння рядків у текстовому режимі. Згідно з цим методом порівняння рядків виконується за абеткою, але без урахування регістра букв. Одже, в текстовому режимі рядок "**ASD"** дорівнює рядку "**asd"**. Для того щоб перейти в текстовий режим порівняння, необхідно на початку модуля перед оголошеннями змінних або процедур помістити оператор **Option Compare Text**.

Для порівняння виразів, які містять посилання на об'єкти, у мові Visual Basic використовується оператор порівняння **Is**. Результатом виконання оператора **Is** є значення **True**, якщо обидва вирази типу **Object** посилаються на один і той самий об'єкт, і значення **False,** якщо посилання стосується інших об'єктів, тобто **obj1 Is obj2**.

Оператор **Like** дозволяє порівнювати рядок із заданим шаблоном і може розглядати тільки два рядкових вирази. Синтаксис цього оператора такий:

StrNGU1 Like StrNGU2.

Тут **StrNGU1** – будь-який рядковий вираз (порівнюваний рядок, у якому виконується пошук), а **StrNGU2** – комбінація спеціальних символів, що визначає шаблон для порівняння. Перелік допустимих символів шаблону з їх коротким описом наведено в табл. 5.4.

Таблиця 5.4

Спеціальні символи шаблону для оператора **Like**

Символ шаблону	Чому відповідає
?	Одному будь-якому символу
*	Будь-якій кількості символів від 0 і більше
#	Будь-якій цифрі від 0 до 9
(List)	Списку (List) певних символів. Для позначення діапазону використовується знак дефіса (-)
(!List)	Будь-якому символу, що не входить у список (List)

Результат порівняння в великих (верхній регістр) і малих (нижній регістр) букв в операторі **Like** залежить від встановленого режиму **Option Compare**. Якщо задано двійкове порівняння рядків, то оператор **Like** розрізняє букви

верхнього й нижнього регістрів. Якщо встановлено режим текстового порівняння, то оператор **Like** не реагує на значення регістрів.

5.5. Строкові оператори

У мові Visual Basic для встановлення рядкових значень використовується єдиний оператор ***конкатенації***, який виконує операцію злиття двох рядків в один. Для позначення такої операції застосовують символ "**&",** або **"+".** Результат конкатенації рядків завжди має тип даних **String**. Наприклад, виконуючи такий оператор:

PIB = "Іванов" & " " & "Іван" " " " " & "Іванович",

у результаті маємо рядок "**Іванов Іван Іванович**".

Якщо в операції конкатенації використовуються числові операнди, то перед її виконанням компілятор Visual Basic автоматично перетворить числові вирази в рядки символів, що відповідають їхньому поточному значенню.

5.6. Пріоритети виконання операцій

Якщо вираз багатоелементний і для його обчислення потрібно більше одного оператора, то цей процес регулюється правилами про пріоритет виконання операцій, що прийняті в мові Visual Basic. Пріоритети основних операторів розглянуто в табл. 5.5. Оператори, що мають однаковий пріоритет, виконуються у виразі послідовно, зліва направо. У мові Visual Basic можлива зміна стандартного порядку виконання операцій у виразах за допомогою дужок.

Таблиця 5.5

Пріоритети операторів Visual Basic

Оператор	Назва	Пріоритет
^	Піднесення до степеня	1
*, /	Множення, ділення	2
\	Ділення без остачі	3
Mod	Остача	4
+, -	Додавання, віднімання	5
&	Злиття рядків	6

Оператори, що мають менше значення пріоритету, виконуються першими. Візьмемо такий наступний приклад:

6 + 7 * 3.

Тут першою буде виконуватись операція множення, оскільки вона має вищий пріоритет, а потім операція додавання. Результат обчислення всього виразу буде дорівнювати 27. При цьому належить зауважити, що коли частини складних виразів взято в дужки, то саме вони завжди обчислюються в першу чергу незалежно від пріоритету операторів.

У прикладі "**(6 + 7) * 3**" першою буде виконано операція додавання, взята в дужки, а тільки потім операція множення. Результат обчислення всього виразу,

буде дорівнювати 39. У виразі можна використовувати скільки завгодно рівнів дужок, але всі вони мають бути парними, тобто відкриватись і закриватись. Якщо дужки у виразі вкладені одна в одну, то обчислення починаються із виразів, які поміщено у внутрішні дужки.

5.7. Математичні функції

Для роботи із числовими значеннями у мові VB існує цілий ряд математичних функцій, які будуть корисні для розв'язування різноманітних обчислювальних задач, як простих, так і достатньо складних. Вони можуть використовуватися не тільки в арифметичних розрахунках, але й для обчислення різноманітних алгебричних і тригонометричних величин. Усі математичні функції, які має у своєму розпорядженні мова Visual Basic, наведено в табл. 5.6.

Таблиця 5.6

Убудовані математичні функції мови Visual Basic

Функція	Призначення
Abs(число)	Обчислює абсолютне значення числа (модуль числа)
Atn(число)	Обчислює арктангенс числа (кут, вимірюваний у радіанах)
Cos(число)	Обчислює косинус числа, що розуміється як кут, виміряний у радіанах
Exp(число)	Обчислює константу **e** в степені, який дорівнює заданому числу (**e**, приблизно дорівнює 2,718282)
Fix(число)	Обчислює цілу частку числа. Для від'ємного числа обчислює найближче більше від'ємне ціле число, або рівніше вказаному
Int(число)	Обчислює цілу частина числа. Для від'ємного числа обчислює найближче менше від'ємне ціле число, або рівніше вказаному
Log(число)	Обчислює натуральний логарифм числа, значення з плаваючою точкою, подвійній точності
Rnd(число)	Генерує випадкове число, значення з плаваючою точкою, одинарній точності

Закінчення табл. 5.6

Sgn(число)	Обчислює знак числа: 1 – число позитивне; 0 – дорівнює нулю; –1 – число від'ємне
Sin(число)	Обчислює синус числа, що розуміється як кут, виміряний у радіанах
Sqr(число)	Обчислює квадратний корінь з числа
Tan(число)	Обчислює тангенс числа, що розуміється як кут, виміряний у радіанах

5.8. Програмування за допомогою процедур і функцій

5.8.1. Характеристика процедур

У практиці програмування часто виникає потреба у виконанні одних і тих самих обчислень, але з використанням різних початкових даних. Щоб уникнути повторення однакових записів і зробити тим самим програму простішою та зрозумілішою, можна виділити повторювані обчислення, в самостійну частину програми, яка при потребі може бути використана багато. Така автономна частина програми, що реалізовує певний алгоритм і допускає звернення до нього з різних частин загальної програми, називається процедурою. Будь-яка процедура містить заголовок і розділ операторів з відомостями про послідовність їх застосування. По суті, процедура дуже схожа на програму. Синтаксис оголошення процедури такий

Sub MyProc(A1, A2, A3, ... ,AN)
[Оператори]
End Sub,

Де **MyProc** – ім'я створюваної процедури;

Оператори – оператори, використовувані в процедурі.

Присвоєння процедурі певного імені повинне відбуватися за певними правилами. У дужках після ім'я процедури роблять перелік формальних параметрів **A1, A2, A3, ... ,AN**, від яких залежатиме результат виконання процедури.

Формальні параметри – це найменування змінних величин, через які інформація з основної програми або іншої процедури передається в виконувану процедуру.

Характеризуючи процедури й функції, слід зазначити, що змінні, які використовуються в програмі, можуть бути локальними й глобальними. Локальні змінні (оголошені тільки в процедурі або функції) існують тільки під час виконання певної процедури або функції. Глобальні змінні (оголошені в

самій програмі) одночасно поширюються на процедури і функції. Такі змінні величини існують, поки програма виконується.

Для того щоб "запустити" процедуру в роботу, необхідно до неї звернутися (тобто викликати її). Виклик процедури відбувається за допомогою операторів виклику, а саме:

MyProc A1, A2, A3 ...,AN

або

Call MyProc (A1, A2, A3 ...,AN)

При цьому список фактичних параметрів може бути відсутнім.

Одночасно діють такі правила зв’язку між фактичними і формальними параметрами процедури:

● кількість фактичних параметрів має дорівнювати кількості формальних параметрів;

● порядок проходження і процедурі й тип фактичних та формальних параметрів мають бути однаковими, але не обов'язково однаково позначені (тобто формальний і фактичний параметри можуть мати різні імена).

Виконання оператора виклику процедури відбувається таким чином:

● усі формальні параметри замінюються відповідними фактичними;

● виникає так званий динамічний екземпляр процедури, який і виконується;

● після виконання процедури відбувається передача керування процесом в основну програму, тобто вступає в дію оператор, наступний за оператором виклику процедури.

Приклад використання процедури в програмі (без врахування параметрів).

Завдання. Вивести на форму значення такого виразу: (4 + 5 + 6) * 200 / 2, використовуючи процедуру Res.

Текст програми:

```
Sub Res
  Print ( (4+5+6)*200/2)
  End Sub
```

Результат обчислення: 1500.

Оголошувати процедуру можна в будь-якій частині програми (на початку, у середині чи в кінці).

Приклад використання процедури в програмі (з урахуванням параметрів).

Завдання. Увести значення трьох змінних за допомогою процедури vvod і роздрукувати їх значення.

Текст програми:

```
option explicit
dim a, b, c    'опис глобальних змінних
Sub Vvod(x)  'процедура введення значень змінних, x – формальний
                    ' параметр
   x=InputBox ("Уведіть значення змінної:")
End Sub
Vvod a→'Звернення до процедури vvod, a – фактичний параметр
Vvod b→'Звернення до процедури vvod, b – фактичний параметр
```

Vvod з→'Звернення до процедури vvod, **с** – фактичний параметр

5.8.2. Характеристика функцій

Функція являє собою процедуру, що обчислює результат. Функція оформляється аналогічно процедурі, але відрізняється від останньої тим, що вона має тільки один результат виконання, який позначається іменем функції і повертається (передається) в основну програму.

Для процедури типу **Function** характерний такий синтаксис:

```
Function Ім'я (аргументи) As тип
  (оператори)
  Ім'я = повертане_значення
End Function
```

Тут ключове слово **Function** визначає процедуру однойменного типу, після нього йде унікальне **Ім'я** процедури-функції, складене за правилами мови Visual Basic. Після імені функції подається перелік її аргументів **аргументи**, який містить передані цей функції дані. **Тип** визначає тип даних величини **повертане значення**, обчислювано функцією в точку виклику за допомогою приміщення її в змінну з ім'ям **Ім'я**. Значення яке повертається функцією, може мати будь-який тип даних, що існує в мові Visual Basic. Коли тип даних цього значення при оголошенні опущений, то передбачають, що це **Variant**.

Після створення функції слід зберегти її в спеціальному модулі для подальшого використання.

Виклик функції відбувається таким чином:

● без присвоєння:

```
Ім'я аргумент1, аргумент2 ..., аргумент n
```

● із присвоєнням:

```
x=Ім’я (аргумент1, аргумент2 ..., аргумент n)
```

Усередині тіла процедури або функції можна оголошувати нові змінні за допомогою ключового слова **Dim**.

Приклад суспільного використання функції та процедури.

Завдання. Визначити відстань, пройдену фізичним тілом, знаючи початкові величини часу, швидкості й прискорення. Використовуємо такий текст програми:

```
Dim v, t, a
Function Rasst(x, y, z)
  Rasst =x*y+z*y*y/2
End Function
Sub Vvod(param, x)
  x = InputBox("Введіть значення параметра param:")
End Sub
Private Sub Commandl_Click()
  Print "Завдання:"
  Print "Визначити відстань, пройдену _ фізичним тілом"
```

```
    "за час t, зі швидкістю v і прискоренням a"
  Vvod "швидкість",v
  Vvod "час", t
  Vvod "прискорення", a
  Print "Тіло пройшла відстань"; Rasst(v, t, a)
End Sub
```

При використанні у фунциях рекурсій використовується стекова пам'ять. (стек).

СТЕК - це послідовна організація пам'яті для структурного даного з доступом до елементу даного, що внесен до структури останнім.

Стек є як би протилежністю черги, оскільки він працює за принципом "останнім прийшов - першим вийшов".

При роботі із стеками операції занесення і витягання елементу є основними. Ці операції традиційно називаються "Заштовхати в стек" і "виштовхнути із стека".

Стек - це робоча область пам'яті, яка динамічно збільшується або зменшується залежно від потреб виконуваної програми. Щоб попередити переповнювання стека необхідно виконати наступні умови:

- переконаєтеся, що функції або процедури не є глибоко вкладеними;
- якщо для локальних змінних недостатньо місця, спробуйте оголосити деякі змінні на рівні модуля. Крім того, усі змінні в процедурі або функції можна оголосити статичними, поставивши попереду ключового слова **Property**, **Sub** або **Function** ключове слово **Static**. Оператор **Static** можна також використовувати для оголошення в процедурах або фунциях окремих статичних змінних;
- перевизначите деякі рядки фіксованої довжини як рядки змінної довжини, оскільки рядки фіксованої довжини використовують більший простір стека в порівнянні з рядками змінної довжини. Крім того, рядок можна визначити на рівні модуля, де їй не вимагається місце в стеку;
- перевірте число вкладених викликів функції **DoEvents** за допомогою діалогового вікна **Calls**, в якому можна проглянути, які процедури використовують стек;
- переконаєтеся, що не був запущений "каскад подій", який є результатом включення події, що викликає процедуру події, яка вже знаходиться в стеку. Каскад подій схожий на виклик незавершеної рекурсивної процедури, проте він не настільки помітний, оскільки виклик виконується системою **Visual Basic** і явним чином не визначений в коді. Для перегляду процедур, діючих в стеку, використовуйте діалогове вікно **Calls**.

6. ПРОЕКТУВАННЯ РОЗГАЛУЖЕНИХ АЛГОРИТМІВ У СЕРЕДОВИЩІ VISUAL BASIC

6.1. Оператор безумовного переходу

Для нього характерний такий синтаксис:

Goto <мітка>.

Тут **мітка** являє собою сукупність букв і цифр кількістю не більшt 40. Крім оператора, мітка ставиться на початку рядка, до якого потрібно перейти, і закінчується вона двокрапкою.

Наприклад:

Goto M1

оператори програми

M1: оператор

6.2. Оператор умовного переходу

Цей оператор має такий синтаксис для різних його типів:

● ***Простий оператор:***

If *умова* **Then** *оператор 1* [**Else** *оператор 2*]

У даному випадку, якщо виконується зазначена умова, то керування процесом передається операторові 1, в інакших випадках – операторові 2.

Приклад:

If x > = 0 Then y = a * x + b Else y = a * x - b ,

де **y**, **a, x, b** – імена змінних.

Примітка. *Я*кщо оператор не поміщається в рядку вікна, то його можна перенести на наступний рядок, використовуючи в місці розриву пропуск і знак підкреслення " _".

● ***Складний оператор:***

If умова 1 **Then**
  [оператори]
 [ElseIf умова 2 **Then**
  [оператори]
[Else [оператори]]
 End If

Складний оператор може включати вкладені умови. Він *завжди записується відповідно до встановленої синтаксисом структури* і закінчується зарезервованим словом **End If.**

Примітка. Якщо кілька операторів в одному рядку відносяться до частин **Then** або **Else,** то вони відокремлюються один від одного двокрапкою.

Сам оператор **If** може вступати в дію у простій або складній формі.

Наприклад,

$$y=\begin{cases} x/a - x/b & якщо\ a \le 5 \quad і \quad x > 6; \\ (a+b)/x & якщо\ a \le 5 \quad і \quad x = 6; \\ a^3 + b * e^{3,5} & в\ інших\ випадках. \end{cases}$$

Виконуючи обчислення за першою формулою, крім зазначених умов, необхідно передбачити, щоб **a** ≠ 0, **b** ≠ 0 . Тоді запис складного оператора буде мати такий вигляд:

```
If  Val(a.Text) <= 5 And Val(x.Text) = 6    Then
    y = (Val(a.Text) + Val(b.Text)) / Val(x.Text)
 ElseIf Val(a.Text) <= 5 And Val(x.Text) > 6 And Val(a.Text) <> 0 And _
   Val(b.Text) <> 0 Then
   y = Val(x.Text) / Val(a.Text) - Val(x.Text) / Val(b.Text)
Else
   y = Val(a.Text) ^3 + Val(b.Text) * exp(2.5)
End If
```

У цьому записі передбачено, що **y**, **a, x, b** являють собою імена текстових елементів (**Textbox**), $\mathbf{e^2}$ – експоненціальну функцію.

6.3. Оператор вибору

Для виконання однієї з багатьох альтернативних дій зручно використовувати ***оператор вибору,*** для якого передбачено такий синтаксис:

```
Select Case <змінна>
   Case порівняння 1
       оператори
  Case порівняння 2
        оператори...
  CaseElse
        [операторі]]  'це необов'язковий оператор конструкції
End select
```

При цьому вибір варіанта залежить від значення керуючої змінної, котре порівнюється з умовами, що подаються після оператора **Case.** Якщо всі порівняння невдалі, то виконуються оператори записані після **Case Else.** Умови, які перебувають у полі порівняння, можуть мати досить різноманітний характер. Наприклад:

```
Select Case alfa
  Case 1
   'оператори відсутні
  Case 2 To 4, 7 To 9,11,13, Is > Max
    'значення alfa порівнюється з 2, 3, 4, 7, 8, 9, 11, 13 і з Max.
    'якщо значення змінної співпаде з яким-небудь
    'числу або виконається порівняння, то відпрацюють два
    'наступних оператори і потім End Select.
    b.Text=alfa + Max
    Printer.Print "b= ", b.Text
  Case Else
    alfa=0
End Select
```

Приклад 6.1

Завдання: спроектувати форму і створити програмний код для обчислення

такої функції: $S = 320 + x * t$, якщо відомі діапазони, в яких змінюються величини x й t: $-750 < x <= 750$ і $-500 < t <= 500$.

У створеному проекті встановити командну кнопку для очищення полів уведення/виведення, передбачити можливість уведення вхідних даних як за допомогою лінійок прокручування (у разі, коли числа цілі й перебувають у зазначених діапазонах), так і безпосередньо через текстові вікна; уставити оператори контролю введення даних користувачем (повинні вводитися тільки числа, в іншому випадку необхідно видати повідомлення про помилку).

Виконання. В екранну форму встановити дві лінійки прокручування для вибору значень змінних ***x*** і ***t***. Послідовність дій, спрямовану на додавання лінійок у форму та встановлення властивостей для них, описано нижче. Контроль уведення даних необхідно виконувати в текстових вікнах ***x*** й ***t.*** Користувач може набирати цифри, знаки + або – , використовувати клавіші **Delete** і **Backspace,** закінчувати введення даних клавішею **Enter, у**сі інші символи повинні зумовлювати повідомлення про помилку. При натисканні на зазначені клавіші на клавіатурі в комп'ютер надходить відповідний код, який можна перевіряти програмно. Оскільки перевіряти потрібно досить велику кількість кодів, то в цьому випадку краще використовувати оператор **Select case.** При цьому спочатку виконують перевірку введення, а потім обчислення, натискуючи клавішу **Enter** на клавіатурі.

Щоб виконати завдання, сформульоване у прикладі 6.1, потрібно провести такі операції:

1. Створити форму. Схему розміщення компонентів на поверхні форми забражено на рис. 6.1.

2. Створити лінійки для вибору значень змінної **x** таким чином:

● на панелі інструментів вибрати елемент керування  – **VscrollBar,** створити за його допомогою лінійку прокручування й розмістити її у формі, як це показано на рис. 6.1 (**vscroll1**);

● активізувати вікно **Properties** і в ньому встановити властивості такі для лінійки:

Name – Vscroll1
Max – 750
Min – –750
Value – 0
SmallChange – 1
LargeChange – 10

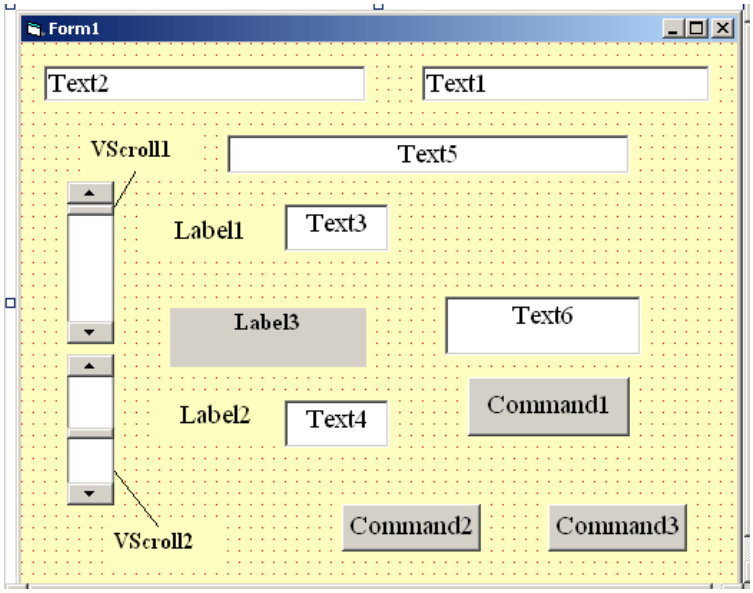


Рис. 6.1. Загальний вигляд початкових даних для створення форми проекту

3. Створити лінійку для вибору значень змінної **t:**

● на панелі інструментів вибрати елемент керування  **VscrollBar,** створити за його допомогою лінійку прокручування і розмістити її у формі, як це показано на рис. 4.1 (**vscroll2**);

● активізувати вікно **Properties** і в ньому встановити властивості такі для лінійки:

Name – vscroll2
Max – 500
Min – –500
Value – 0
SmallChange – 1
LargeChange – 6

4. Внести таки доповнення в програмний код:

● подвійним клацанням лівою клавішею миші по верхній (у формі проекту) лінійці прокручування відкрити вікно **Code;**

● для об'єкта ***x*** вибрати подію **Change** і дописати процедуру, виконання, якої буде необхідним після настання події, тобто переміщення бігунка на лінійці із контролем уведених даних:

```
Private Sub VScroll1_Change()
  Text3.Text = VScroll1
  mes.Caption = "Уведення змінної a в діапазоні –750...750"
End Sub;
```

● для об'єкта ***x*** вибрати подію **Scroll** і дописати зміст процедури, виконуваної під час переміщення бігунка по лінійці, а саме:

```
Private Sub VScroll1_Scroll()
  VScroll1_Change
End Sub;
```

● аналогічно створити процедури для об'єкта ***t***, тобто:

```
Private Sub VScroll2_Change()
  Text4.Text = VScroll2
  mes.Caption = "Уведення змінної a в діапазоні -500...500"
End Sub;
Private Sub VScroll2_Scroll()
  VScroll2_Change
End Sub
```

5. Скористатись відомостями про призначення всіх інструментів, які розміщені на поверхні форми, за табл. 6.1.

6. Додати оператори контролю введення в код текстового вікна. Подвійним клацанням лівою клавішею миші по текстовому вікну **text3** викликати вікно коду величини ***x***, та ***y*** ньому відкрити подію **KeyPress**. При цьому з'являється заготівка процедури:

```
Sub text3_KeyPress (KeyAscii As Integer)
End Sub
```

Таблиця 6.1

Послідовність дій для створення форми проекту

Операція	Властивість об'єкта	Значення властивості
1. Дати ім'я електронній формі	**Caption**	**Form1**
2. Створити лінійку прокручування для вибору значень змінної ***x*** (інструмент **VScrollBar)**	**Name**	**VScroll1**
3. Створити лінійку прокручування для вибору значень змінної ***t*** (інструмент **VScrollBar)**	**Name**	**Vscroll2**
4. Створити вікно для відображення назви вищого навчального закладу (інструмент **TextBox)**	**Name**	**Text2**
	Text	Національний гірничий університет
5. Створити вікно для відображення назви групи та прізвища студента (інструмент **TextBox)**	**Name**	**Text1**
6. Створити вікно для відображення назви групи та прізвища студента (інструмент **TextBox)**	**Text**	Назва групи та прізвище студента

Закінчення табл. 6.1

7. Створити вікно для уведення та відображення змінної ***x*** (інструмент **TextBox)**	**Name**	**Text3**
8. Створити вікно для уведення та відображення змінної ***t*** (інструмент **TextBox)**	**Name**	**Text4**
9. Створити вікно для відображення обчислювальної формули (інструмент **TextBox)**	**Name**	**Text5**
10. Створити вікно для відображення результатів розрахунку (інструмент **TextBox)**	**Name**	**Text6**
11. Створити підказку для введення змінної ***x*** (інструмент **Label)**	**Name**	**Label1**
	Caption	**Уведення *x***
12. Створити підказку для введення змінної ***t*** (інструмент **Label)**	**Name**	**Label2**
	Caption	**Уведення *t***
13. Створити підказку для контролю введення даних змінних ***x*** і ***t*** (інструмент **Label)**	**Name**	**Mes**
	Caption	Стерти назву **Label3**
14. Створити командну кнопку для виконання розрахунку за обраної формулою (інструмент **CommandButton)**	**Name**	**Command1**
	Caption	**Результат S**
15. Створити командну кнопку для очищення полів **x**, **t, S, Mes** (інструмент **CommandButton)**	**Name**	**Command2**
	Caption	**Очищення**
16. Створити командну кнопку для виходу з програми (інструмент **CommandButton)**	**Name**	**Command3**
	Caption	**Вихід**

Оголошений автоматично параметр **KeyAscii** служить для прийняття коду натиснутої на клавіатурі клавіші.

Додати в заготівку оператор контролю введення таким чином:

```
Private Sub text3_KeyPress(KeyAscii As Integer)
  Select Case KeyAscii
    Case 0, 8, 13, 45, 46, 48 To 57
    '0,8 - Delete, BackSpace, 13 - Enter
    '45 – "–", 46 – ".", 48–57 – цифри від 0 до 9
  Case Else
     KeyAscii = 0
     mes.Caption = "Помилка уведення – Уведить число"
  End Select
End Sub.
```

7. Додати оператори контролю введення в код текстового вікна **text4**. Для текстового вікна **text4** усі дії виконуються аналогічно вікна **text3.** У вікні виділити рядки від **Select case** до **End select**, виконавши потім такі дії:

скопіювати за допомогою пункту меню **Edit**→ команда **Copy**;

відкрити вікно коду текстового вікна **text4**, встановити подію **KeyPress**, а потім – пункт меню **Edit** → команду **Paste.**

Остаточний вигляд екранної форми проекту для виконання завдання прикладу 6.1 подається на рис. 6.2.

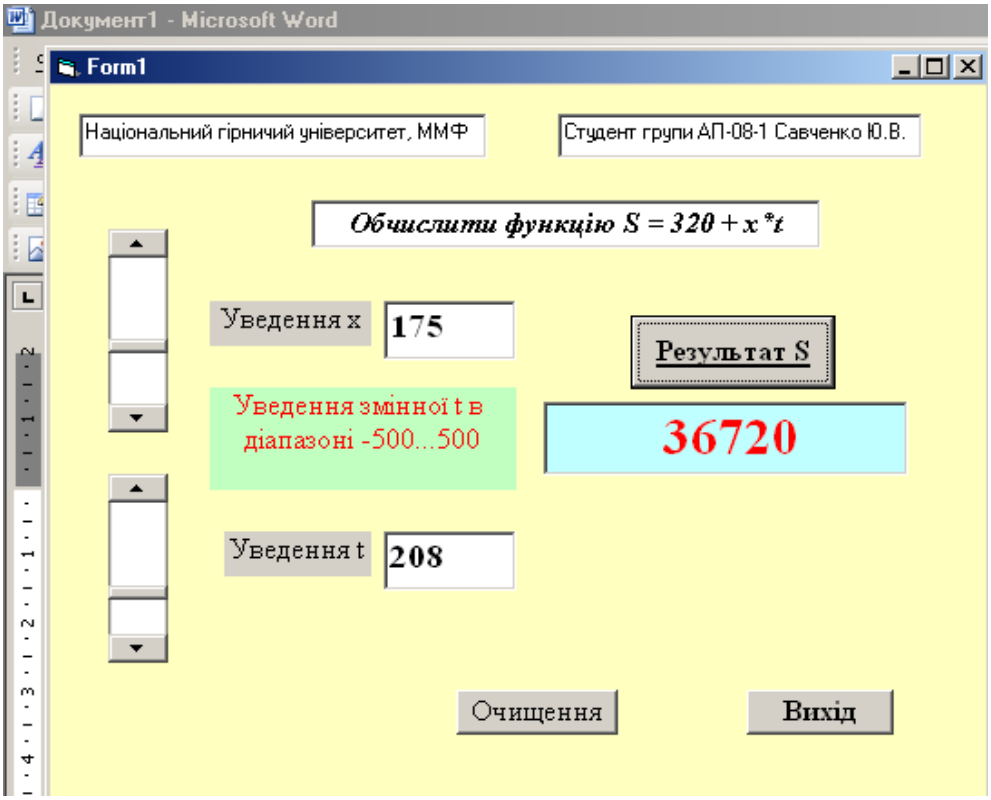


Рис. 6.2. Остаточний вигляд екранної форми виконання завдання з прикладу 6.1

8. Відобразити повний програмний код:

```
' Процедура очищення текстових вікон виведення результатів
Private Sub Command2_Click()
  Text3.Text = ""
  Text4.Text = ""
  Text6.Text = ""
  mes.Caption = ""
End Sub
Private Sub VScroll2_Change()
  Text4.Text = VScroll2
  mes.Caption = "Уведення змінної t в діапазоні –500...500"
End Sub

' Контроль уведення даних
Private Sub text3_KeyPress(KeyAscii As Integer)
  Select Case KeyAscii
   Case 0, 8, 13, 45, 46, 48 To 57
   '0,8 – Delete, BackSpace, 13 – Enter
```

```
   '45 – "–", 46 – ".", 48–57 – цифри від 0 до 9
  Case Else
    KeyAscii = 0
    mes.Caption = "Помилка введення – Уведіть число"
  End Select
End Sub

' Контроль уведення даних
Private Sub text4_KeyPress(KeyAscii As Integer)
  Select Case KeyAscii
   Case 0, 8, 13, 45, 46, 48 To 57
    '0,8 - Delete, BackSpace, 13 - Enter
    '45 - "-", 46 - ".", 48-57 - цифри від 0 до 9
  Case Else
    KeyAscii = 0
    mes.Caption = "Помилка введення – Уведіть число"
  End Select
End Sub
```

9. Виконати створену програму, вибираючи значення для змінних величин **x** і **t** шляхом переміщення бігунків на лінійках прокручування.

Приклад 6.2

Завдання: - Розробити й налагодити програму для обчислення функції **Z** у такому розгалуженому процесі:

$$Z = \begin{cases} xy - xy^2, & якщо\ x \le 2\ \ i\ \ y \le 6; \\ \dfrac{(x+y)^2}{x^2+1}, & якщо\ x \ge 0\ \ i\ \ y \ge 2; \\ x^3 + y + e^{2,5x}, & в\ \ інших\ \ випадках. \end{cases}$$

Виконання: Спочатку необхідно спроектувати електронну форму завдання (див. рис. 6.3).

Занесення формули рекомендується здійснювати в текстовому редакторі *Word,* а потім скопіювати в керуючий елемент **OLE.**

Послідовність операцій для виконання завдання, сформульованого в прикладі 4.2 така:

1. Створити електронну форму у проекту і задати властивості її елементів таким чином:

● запровадити дії аналогічні записаним у попередніх прикладах, при цьому імена компонентів полів відображено на рис. 6.3.

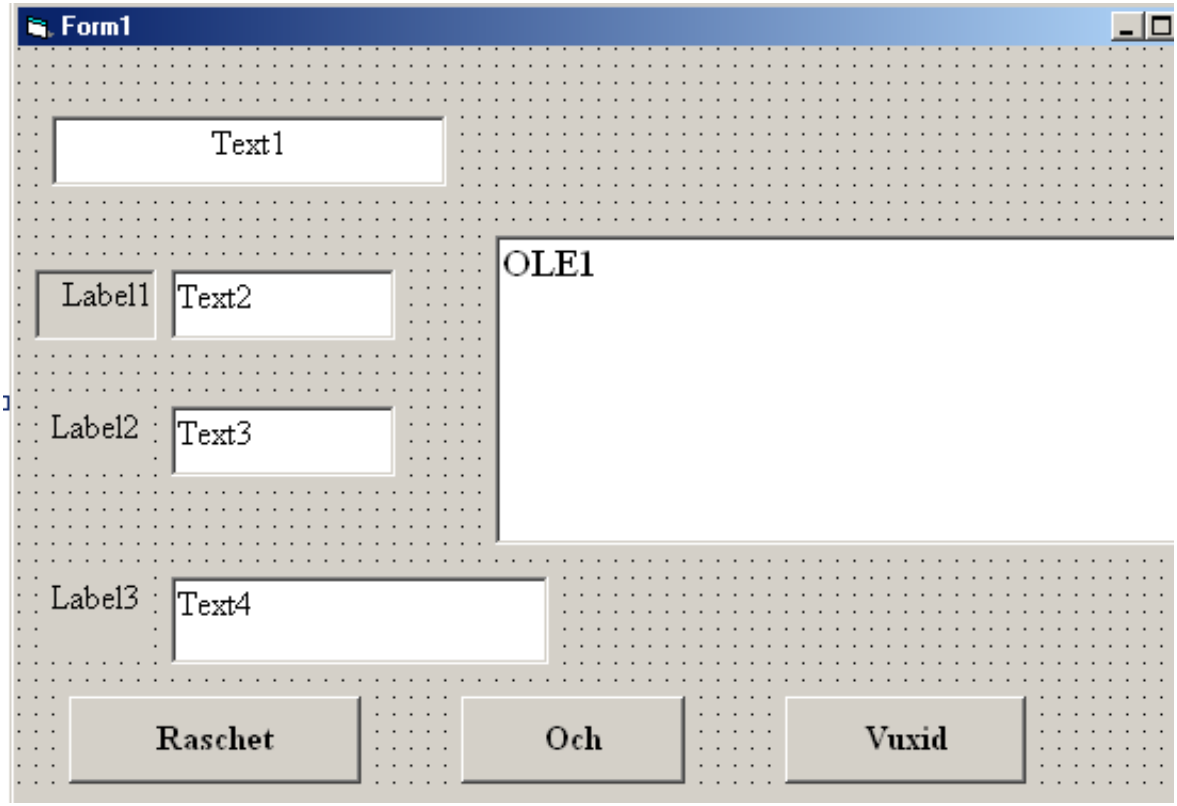


Рис. 6.3. Початкові дані для створення форми з метою виконання завдання, сформульованого в прикладі 6.2

● створити форму **Складне розгалуження,** виконавши послідовно операції, перелічені в таблиці 6.2:

Таблиця 6.2

Послідовність операцій з метою створення форми – **Складне розгалуження**

Операція	Властивість об'єкта	Значення властивості
1. Дати ім'я електронній формі	**Name**	***Form1***
	Caption	***Складне розгалуження***

Закінчення табл. 6.2

2. Створити текстове вікно для відображення номера академічної групи і прізвища студента (інструмент **TextBox)**	**Name**	***Text1***
	Text	***Проект розробив студент гр. АП-08-1***
	Multline	***True***
3. Створити текстове вікно для введення змінної **x** (інструмент **TextBox)**	**Name**	***Text2***
	Text	порожньо
4. Створити текстове вікно для введення змінної **y** (інструмент **TextBox)**	**Name**	***Text3***
	Text	порожньо
5. Створити текстове вікно для виведення змінної **z** (інструмент **TextBox)**	**Name**	***Text4***
	Text	порожньо
6. Створити етикетку для ідентифікації змінної **x** (інструмент **Label)**	**Label1**	**x =**
7. Створити етикетку для ідентифікації змінної **y** (інструмент **Label)**	**Label2**	**y =**
8. Створити етикетку для ідентифікації змінної **z** (інструмент **Label)**	**Label3**	**z=**
9. Створити командну кнопку для виконання розрахунку функції **z** (інструмент **CommandButton)**	**Name**	***Raschet***
	Caption	***Розрахунок***
10. Створити командну кнопку для очищення полів **x** і **y** (інструмент **CommandButton)**	**Name**	***Och***
	Caption	***Очищення***
11. Створити командну кнопку для виходу з програми (інструмент **CommandButton)**	**Name**	***Vuxid***
	Caption	***Вихід***
12. Створити вікно для відображення функції **z** (інструмент OLE)	**Name**	**OLE1**

2. Необхідний програмний код:

```
' Процедура розрахунку функції z
Private Sub Raschet_Click()
  ' Оголошення типу даних
    Dim x, y, z As Single
```

```
    ' Перетворення рядкових даних у числові
     x = Val(Text2.Text)
     y = Val(Text3.Text)
     z = Val(Text4.Text)

    ' Вибір функції залежно від значення змінних x і y
    If x <= 2 And y >= 6 Then
        z1 = x * y - x * y ^ 2
        Text4.Text = Str(Format$(z1, "0.00"))
    ElseIf x >= 0 And x >= 2 Then
       z1 = (x + y) ^ 2 / (x ^ 2 + 1)
       Text4.Text = Str(Format$(z1, "0.00"))
    Else
        z1 = x ^ 3 + y * Exp(2.5 * x)
        Text4.Text = Str(Format$(z1, "0.00"))
    End If
End Sub

' Процедура виходу з програми
Private Sub Vuxid_Click()
  End
End Sub

' Процедура очищення текстових вікон
Private Sub Och_Click()
  Text2.Text = ""
  Text3.Text = ""
  Text4.Text = ""
End Sub
```

Остаточний вигляд екранної форми проекту для виконання завдання з прикладу 6.2 подається на рис. 6.4.

3. Виконати створену програму, для чого необхідно спочатку ввести початкові дані: змінну **x** у вікно з підказкою "**X=**", а змінну **Y** у вікно з підказкою "**Y=**", а потім клацнути лівою клавішею миші у місці кнопки "**Розрахунок**". Результат розрахунку функції **z** виводиться у вікні з підказкою "**Z=**".

6.4. Селекторні кнопки (перемикачі), прапорці, рамки

Опис роботи при виконанні попередніх прикладів показав, що користувач працював "наосліп", тобто уводив дані й одержував відповідь. Усе, про що його інформувала програма – це умова завдання. Можна зробити користувача більш активним учасником процесу, наприклад, давати йому можливість вибирати потрібну формулу для розрахунку або задавати кілька необхідних параметрів

задачі. Для цього на першому етапі розробки проекту варто передбачити використання таких керуючих елементів, як **селекторні кнопки (перемикачі), рамки, прапорці,** причому їх можна помістити у форму за допомогою таких елементів панелі інструментів:

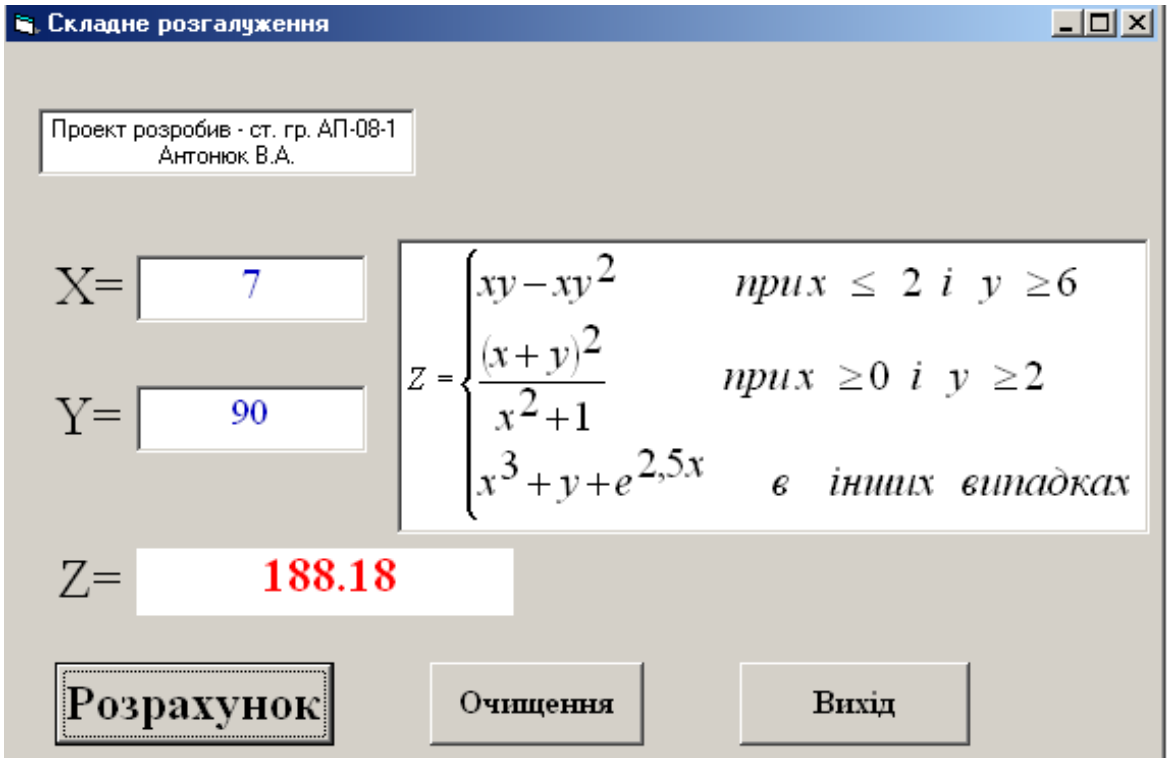


Рис. 6.4. Остаточний вигляд екранної форми для виконання завдання з прикладу 6.2

– селекторна кнопка (перемикач) **(OptionButton)**

– рамка **(Frame)**

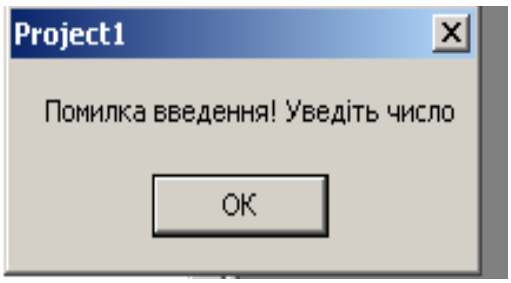
– прапорець **(CheckBox)**

Крім того, працюючи в середовищі Windows, користувач звик одержувати повідомлення від машини у вигляді інформаційних вікон. Це можна також здійснити і в середовищі Visual Basic за допомогою функції **MsgBox.**

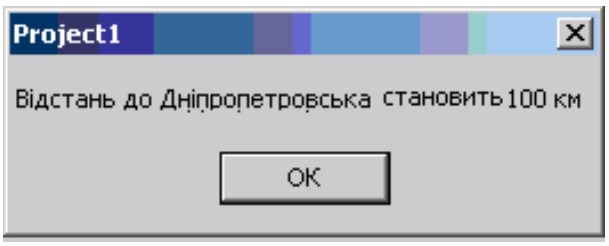
Мінімальний формат цієї функції такий:

MsgBox "текст повідомлення"

У текст повідомлення можна включати кілька текстових констант і змінних, об'єднуючи їх за допомогою операції конкатенації (+ або &). Якщо необхідно включити числову змінну, то її варто перетворити в рядковий тип даних за допомогою функції **Str** (змінна). Наприклад, унаслідок виконання функції **MsgBox "Помилка введення! Уведіть число"** на екрані з'явиться вікно (рис. 4.4, *а*), а в результаті виконання функції **MsgBox "Відстань до Дніпропетровська становить ." & Str(P) & "км"**– інформаційне вікно (рис. 6.5, *б*). **P** – змінна, яка включає відстань до Дніпропетровська.



а



б

Рис. 6.5. Інформаційні вікна, отримані за допомогою виконання функції **MsgBox**

Вибір одного з альтернативних рішень здійснюється за допомогою селекторних кнопок, коли в будь-який момент можна натиснути тільки одну з них. Для селекторної кнопки існує властивість **Value,** що відоббражає значення даної кнопки за умовчуванням, а саме:

– ***увімкнуто.*** Для встановлення такої позиції необхідно, щоб значення властивості **Value** дорівнювало **True (Істина).**

– ***вимкнуто.*** Для встановлення такої позиції необхідно, щоб значення властивості **Value** дорівнювало **False (Неправда).**

Аналізуючи значення властивості **Value,** програма вибирає формулу (задачу) для розв'язку.

Щоб об'єднати й ідентифікувати кілька селекторних кнопок у групу, варто спочатку створити в межах форми їхню групову рамку за допомогою елемента панелі інструментів **Frame.** Допускається також розміщення перемикачів прямо у формі, без їхнього об'єднання в групу.

Приклад 6.3

Завдання: обчислити значення функції за однією із трьох формул:

$\mathbf{y = e^x + x^2}$, $\mathbf{y = x^4 + x^2 + 1}$ або $\mathbf{y = x^2 - 1}$, якщо $0 < x \le 100$.

Вибираючи формули для розрахунку, застосовують перемикачі. Значення змінної **x** задають за допомогою лінійки прокручування.

Виконання:

Для обчислення значення функції необхідно здійснити описані нижче операції.

1. Створити екранну форму (див. рис. 6.6).

2. Використовуючи оператора **Print**, помістити на поверхню форми у її лівому верхньому куті прізвище студента, його ініціали і назву групи, в якій він вчиться. Для виконання цієї дії застосувати такий програмний код:

```
Private Sub Form_Load()
  Print "Виконав студент"
  Print "групи АП-08-1"
  Print "Шевченко  Ю. В."
End Sub
```

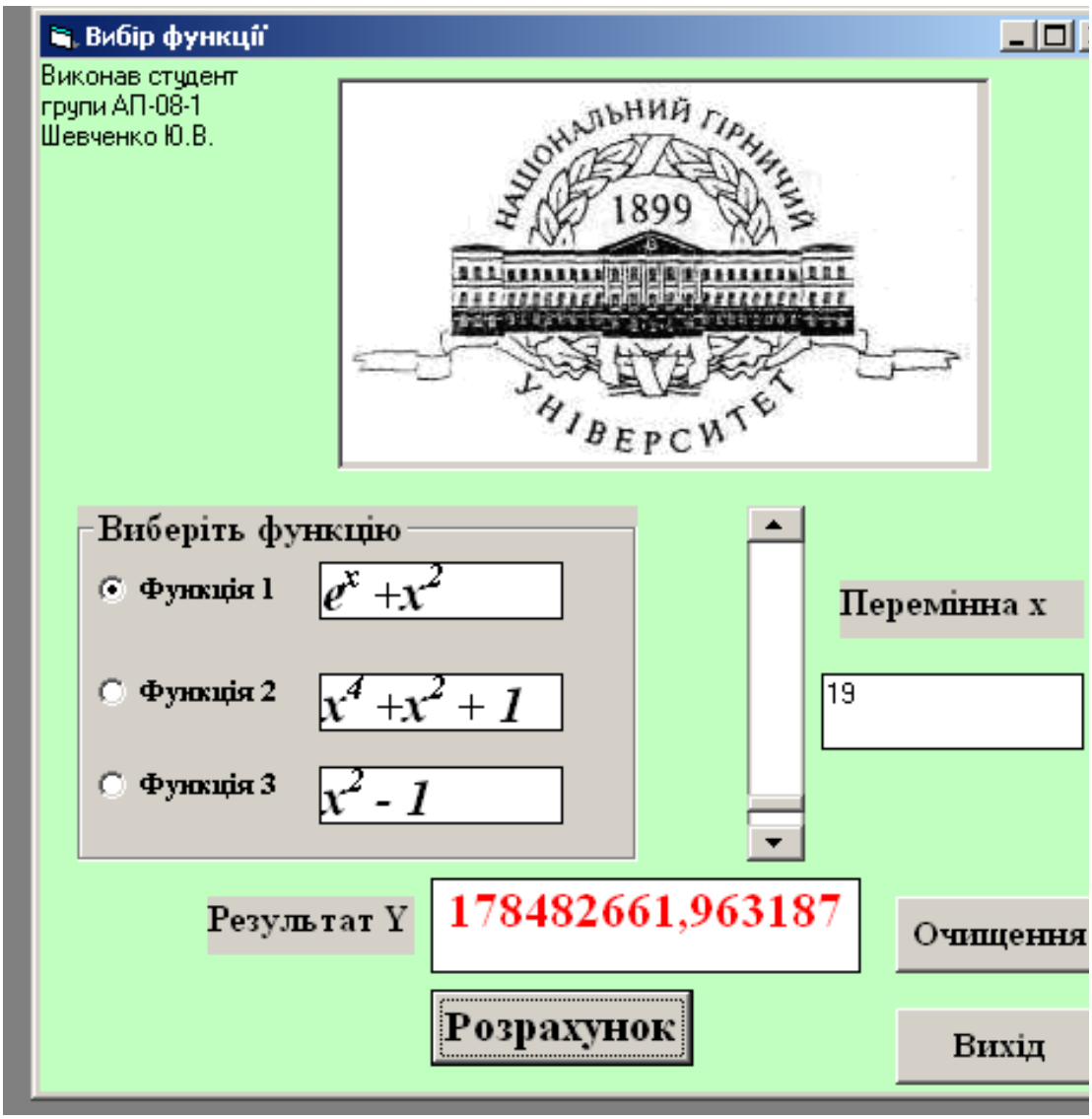


Рис. 6.6. Вигляд екранної форми для виконання завдання з прикладу 6.3

Створити форму **Вибір функції**, виконавши послідовно операції, перелічені в табл.6.3.

Таблиця 6.3

Послідовність операцій для створення форми **Вибір функції**

Операція	Властивість об'єкта	Значення властивості
1. Дати ім'я електронній формі	**Caption**	**Вибір функції**
2. Створити лінійку прокручування для вибору значень змінної **x** (інструмент **HScrollBar)**	**Name**	**LX**
3. Створити рамку для групи перемикачів (інструмент **Frame)**	**Caption**	**Виберіть функцію**

Закінчення табл. 6.3

4. Створити перемикач для вибору функції 1 (інструмент **OptionButton)**	**Name**	**F1**
5. Створити перемикач для вибору функції 2 (інструмент **OptionButton)**	**Name**	**F2**
6. Створити перемикач для вибору функції 3 (інструмент **OptionButton)**	**Name**	**F3**
7. Створити для ілюстрації розрахункову формулу **y** = x / **6** (інструмент **OLE)**	**Name**	**OLE1**
8. Аналогічно п.7 створити дві інші формули	**Name**	**OLE2**
9. Створити етикетку для ідентифікації поля зі значенням змінної x (інструмент **Label)**	**Name**	**Label1**
10. Створити поле для перегляду значення змінної **x** (інструмент **Text)**	**Name**	**X**
11. Створити етикетку для ідентифікації поля результату розрахунків змінної **y** (інструмент **Label)**	**Name**	**Label2**
12. Створити поле для перегляду значення змінної **y** (інструмент **Text)**	**Name**	**У**
13. Створити командну кнопку для виконання розрахунку за вибраної формулою (інструмент **CommandButton)**	**Name**	**Raschet**
	Caption	**Розрахунок**
14. Створити командну кнопку для очищення полів, що містять значення змінних **x** і **y** (інструмент **CommandButton)**	**Name**	**Och**
	Caption	**Очищення**
15. Створити командну кнопку длявиходу з програми (інструмент **CommandButton)**	**Name**	**Vuxid**
	Caption	**Вихід**

У верхній частині форми посередині помістити емблему Національного гірничого університету. Растрове зображення емблеми вставляється у вікно, що створюється інструментом  **PictureBox**.

3. Створити програмний код, виконуючи такі операції:

● Подвійним клацанням лівою кнопки миші у місці лінійки прокручування у формі проекту відкрити вікно **Code.**

Для об'єкта **LX** вибрати подію **Change** і дописати процедуру, яку треба виконати в разі здійснення події – переміщення бігунка на лінійці, а саме:

Private Sub LX_Change()

```
  X.Text = LX.Value
End Sub
```

● Для об'єкта **LX** вибрати подію **Scroll** і дописати зміст процедури, виконуваної під час переміщення бігунка по лінійці, тобто:

```
Private Sub LX_Scroll()
  LX_Change
End Sub
```

● Вибрати в лівому списку об'єкт **Розрахунок,** а в правому – подію **Click.** Описати послідовність операцій, які виконуються в програмі внаслідок клацання лівою клавішею миші у місці кнопки **"Розрахунок"**, а саме:

```
Private Sub Raschet_Click()
  If F1.Value = True Then
    Y.Text = Exp(Val(x.Text)) + (Val(x.Text)) ^ 2
  ElseIf F2.Value = True Then
    Y.Text = (Val(x.Text)) ^ 4 + (Val(x.Text)) ^ 2 + 1
  Else
    Y.Text = (Val(x.Text)) ^ 2 - 1
  End If
End Sub
```

● Вибрати в лівому списку об'єкт **Очищення,** а в правому – подію **Click.** Описати послідовність дій, виконуваних у програмі внаслідок клацання мишкою у місці кнопки **"Очищення"**, тобто:

```
Private Sub Och_Click()
   x.Text = ""
   y.Text = ""
End Sub
```

● Вибрати в лівому списку об'єкт **Вихід,** а в правому – подію **Click.** Описати послідовність дій, виконуваних у програмі внаслідок клацання мишкою у місці кнопці **"Вихід"**.

4. Відобразити повний програмний код:

```
' Створення процедури виходу з програми
Private Sub vuxid_Click()
  End
End Sub


' Створення процедури введення прізвища й назвиі групи, в якій вчиться студент
Private Sub Form_Load()
  Print "Виконав студент"
  Print "групи АП-08-1"
  Print "Шевченко Ю.В."
End Sub
' Створення лінійки прокручування початкового значення x
Private Sub LX_Change()
   x.Text = LX.Value
```

```
End Sub

Private Sub LX_Scroll()
  LX_Change
End Sub

' Очищення текстових вікон виведення результатів
Private Sub Och_Click()
 X.Text = ""
 Y.Text = ""
End Sub

' Розрахунок значення функції Y
Private Sub Raschet_Click()
  If F1.Value = True Then
     Y.Text = Exp(Val(x.Text)) + (Val(x.Text)) ^ 2
   ElseIf F2.Value = True Then
     Y.Text = (Val(x.Text)) ^ 4 + (Val(x.Text)) ^ 2 + 1
   Else
     Y.Text = (Val(x.Text)) ^ 2 - 1
   End If
End Sub

' Вихід з програми
Private Sub vuxid_Click()
  End
End Sub
```

Приклад 6.4

Завдання: автомобіль рухається по дорозі із швидкістю **V** км/год. Через певні обставини водій змушений виконати екстрене гальмування.

Розрахувати гальмівний шлях автомобіля залежно від стану дорожнього покриття, який зумовлюється величиною коефіцієнта зчеплення **f** коліс автомобіля з дорогою.

Для введення швидкості руху автомобіля використати в програмі лінійку прокручування, а для вибору коефіцієнта зчеплення **f** застосувати перемикачі.

Виконання:

1.Створити екранну форму відповідно до рис. 6.7.

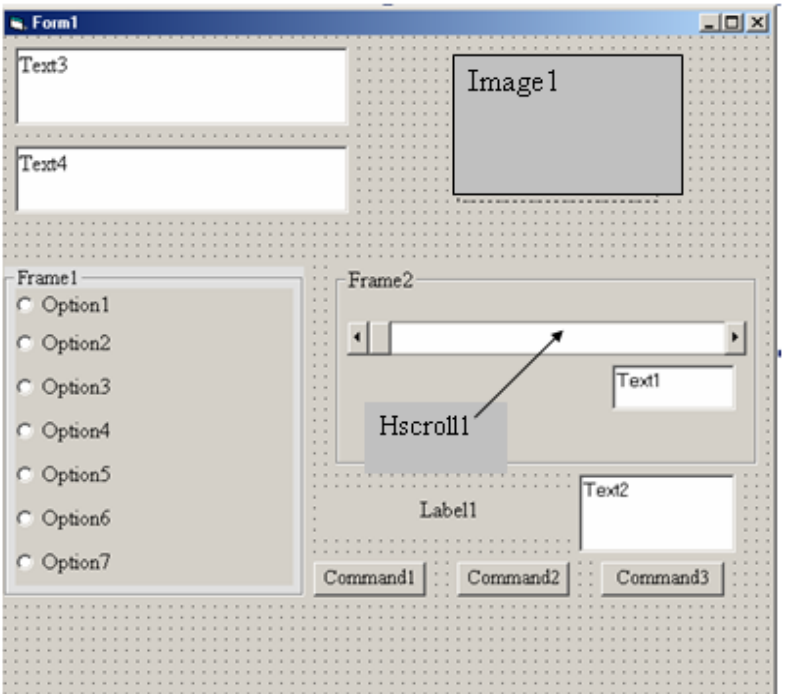


Рис. 6.7**.** Початкові дані для створення форми з метою виконання завдання з прикладу 6.4

Таблиця 6.4

Послідовність операції для створення форми **Вибір функції**

Операція	Властивість об'єкта	Значення властивості
1. Дати ім'я електронній формі	**Name**	***Form1***
	Caption	***Гальм_шлях***
2 Створити текстове вікно для відображення значень гальмівного шляху	**Name**	***Text3***
	Text	***Розрахунок гальмівного шляху автомобіля***
	Multline	***True***
3. Створити текстове вікно для відображення значень гальмівного шляху	**Name**	***Text4***
	Text	Прізвище студента, група
	Multline	***True***
4. Створити рамку для групи перемикачів (інструмент **Frame)**	**Name**	***Frame1***
	Caption	***Вибір коефіцієнта зчеплення f***
5. Створити перемикач для вибору значення коефіцієнта ***f*** = 0,1 (інструмент **OptionButton)**	**Name**	***Option1***
	Caption	***f=0,1***

Продовження табл.6.4

6. Створити перемикач для вибору значення коефіцієнта ***f*** = 0,2 (інструмент **OptionButton)**	**Name**	***Option2***
	Caption	***f=0,2***
7. Створити перемикач для вибору значення коефіцієнта ***f*** = 0,3 (інструмент **OptionButton)**	**Name**	***Option3***
	Caption	***f=0,3***
8. Створити перемикач для вибору значення коефіцієнта ***f*** = 0,4 (інструмент **OptionButton)**	**Name**	***Option4***
	Caption	***f=0,4***
9. Створити перемикач для вибору значення коефіцієнта ***f*** = 0,5 (інструмент **OptionButton)**	**Name**	***Option5***
	Caption	***f=0,5***
10. Створити перемикач для вибору значення коефіцієнта ***f*** = 0,6 (інструмент **OptionButton)**	**Name**	***Option6***
	Caption	***f=0,6***
11. Створити перемикач для вибору значення коефіцієнта ***f*** = 0,7 (інструмент **OptionButton)**	**Name**	***Option7***
	Caption	***f=0,7***
12. Створити рамку для компонентів уведення значення швидкості руху автомобіля (інструмент **Frame)**	**Name**	***Frame2***
	Caption	***Введення швидкості руху автомобіля***
13. Створити лінійку прокручування для вибору значень змінної **V** (інструмент **HScrollBar)**	**Name**	***HScroll1***
	Large Change	***1***
	Max	***250***
	Min	***0***
14. Створити текстове вікно для відображення введених значень швидкості автомобіля	**Name**	***Text1***
	Text	порожньо
	Font	Шрифт – ***Times New Roman*** Написання ***Жирний курсив*** Розмір **–** ***12***

Закінчення табл.6.4

15. Створити текстове вікно для відображення значень гальмівного шляху	**Name**	***Text2***
	Text	порожньо
	Font	Шрифт-***Times New Roman*** Написання ***Жирний курсив*** Розмір – 16
16. Створити етикетку для ідентифікації поля із значенням гальмівного шляху (інструмент **Label)**	**Label1**	***Гальмівний шлях =***
17. Створити командну кнопку для виконання розрахунку гальмівного шляху (інструмент **CommandButton)**	**Name**	***Command1***
	Caption	***Розрахунок***
18. Створити командну кнопку для очищення полів, що містять значення змінних **x** та **y** (інструмент **CommandButton**)	**Name**	***Command2***
	Caption	***Очищення***
19. Створити командну кнопку для виходу з програми (інструмент **CommandButton)**	**Name**	***Command3***
	Caption	***Вихід***

2. Праворуч у верхній частині форми помістити малюнок автомобіля з колекції кліпів текстового процесора Microsoft Word, вводячи в дію команду меню **Вставка→Рисунок→Картинки**. Зображення вставляється у вікно, що створюється компонентом  **Image.**

3. Створити програмний код, який передбачає виконання перелічених нижче процедур:

```
' Процедура створення лінійки прокручування для введення значень
'швидкості автомобіля (використання компоненту HScrollBar)
Private Sub HScroll1_Change()
    Text1.Text = HScroll1.Value
End Sub
Private Sub HScroll1_Scroll()
  HScroll1_Change
End Sub
' Процедура очищення текстових вікон
Private Sub Command2_Click()
  Text1.Text = ""
  Text2.Text = ""
End Sub
' Процедура обчислення гальмівного шляху з використанням
' компонента CommandButton
Private Sub Command1_Click()
  If Option1.Value = True Then    ' вибір коефіцієнта зчеплення з
```

```
' використанням компонента OptionButton
    Text2.Text = (Val(Text1.Text)) ^ 2 * 0.0039327 / 0.1
  ElseIf Option2.Value = True Then
    Text2.Text = (Val(Text1.Text)) ^ 2 * 0.0039327 / 0.2
  ElseIf Option3.Value = True Then
    Text2.Text = (Val(Text1.Text)) ^ 2 * 0.0039327 / 0.3
  ElseIf Option4.Value = True Then
    Text2.Text = (Val(Text1.Text)) ^ 2 * 0.0039327 / 0.4
  ElseIf Option5.Value = True Then
    Text2.Text = (Val(Text1.Text)) ^ 2 * 0.0039327 / 0.5
  ElseIf Option6.Value = True Then
    Text2.Text = (Val(Text1.Text)) ^ 2 * 0.0039327 / 0.6
  ElseIf Option7.Value = True Then
    Text2.Text = (Val(Text1.Text)) ^ 2 * 0.0039327 / 0.7
  End If
End Sub
' Процедура виходу з програми
Private Sub Command3_Click()
  End
End Sub
```

4. Остаточний вигляд форми для розрахунку гальмівного шляху показано на рис. 6.8.

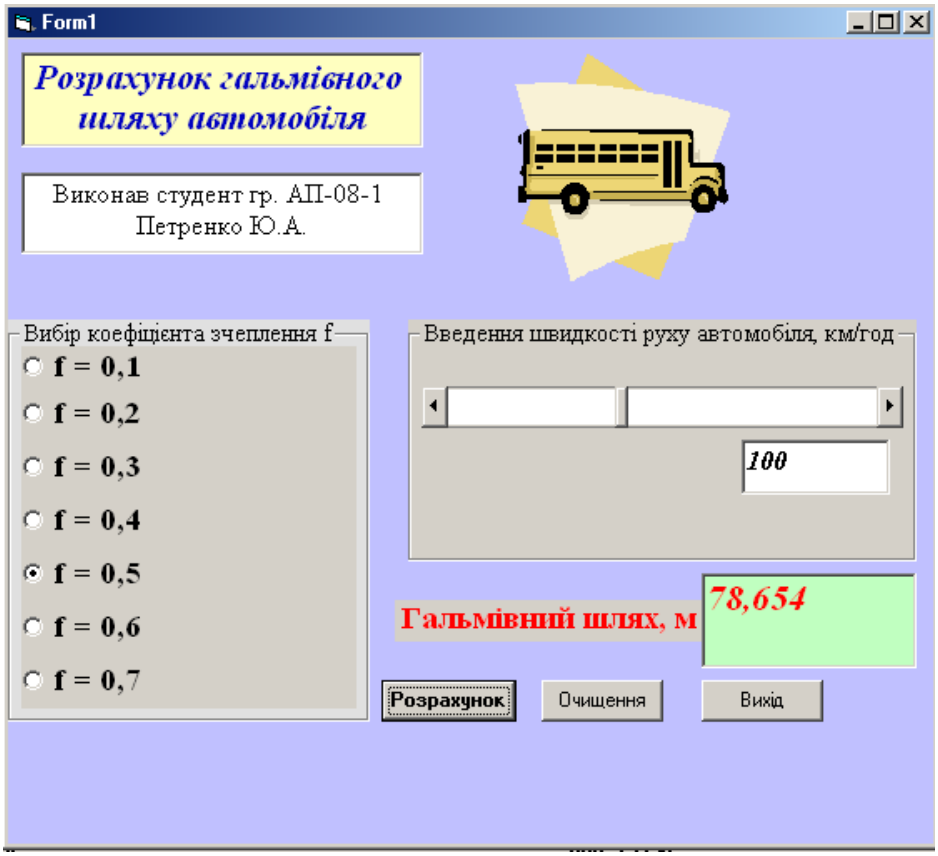


Рис. 6.8. Остаточний вигляд екранної форми для виконання завдання з прикладу 6.4

5. Для розрахунку гальмівного шляху необхідно за допомогою лінійки прокручування ввести значення швидкості руху автомобіля (у даному прикладі

це 100 км/год) і перемикачем вибрати коефіцієнт зчеплення **f** коліс автомобіля з дорожнім покриттям (**f = 0,5**). Натиснення на кнопку **Розрахунок** приведе до появи в текстовому вікні значення гальмівного шляху (78,654 м).

Контрольні питання

1. Дайте визначення розгалуженому процесу обчислень?
2. Яка кількість розгалужень допускається в розгалужених алгоритмах?
3. Якими блоками позначається перевірка умов у розгалужених алгоритмах, які їхні особливості?
4. Які оператори мови характерні саме для розгалужених алгоритмів, яким чином вони працюють?
5. Чим відрізняється складний оператор **If... Then ... Else** від простого?
6. Що означає неповний оператор **If...?**
7. З якою метою використовуються селекторні кнопки в програмах, як задати їм початкове значення?
8. Яку функцію можуть виконувати прапорці в програмах? Наведіть приклади.
9. Як змусити ПК видавати повідомлення у вигляді інформаційних вікон?
10. Для чого в програмах використовуються константи?
11. Спроектувати програму на в середовищі Visual-Basic, використовуючи такі значення змінних величин:

а) $y = \begin{cases} a*c+8{,}5*x, & якщо \quad x>7; \\ a*c, & якщо \quad x=7; \\ (a*c-b)/5*x, & якщо \quad x<7. \end{cases}$

б) $y = \begin{cases} 5*x^2-4*x+5, & якщо \quad x=1; \\ 0{,}5*e^x-(x-1)^2, & якщо \quad x=2; \\ 625 & в \quad інших \quad випадках. \end{cases}$

7. ПРОЕКТУВАННЯ ЦИКЛІЧНИХ ПРОЦЕСІВ

7.1. Загальні положення

Як було розглянуто в п. 3.3.4, циклічні програми можуть бути різних видів – арифметичні, ітераційні, складні, з індексованими змінними. Для їхнього написання використовуються різні оператори.

7.2. Арифметичні цикли

Арифметичні цикли являють собою фрагменти програми, у яких оператори повторюються заздалегідь відому кількість разів, а для їхнього запису використовується такий оператор:

For змінна = початкове значення **To** кінцеве значення **[Step...]**
оператори **[ExitFor]** оператори

Next [змінна [,змінна]...]

Відповідно до цього оператора змінній буде присвоєно початкове значення, виконано перевірку кінця циклу, а якщо кінця немає, то будуть виконані оператори циклу **Next**, при цьому значення змінної буде збільшено на значення кроку (**Step**) або, якщо останній не передбачено то на одиницю, потім операція повторюється, починаючи з процесу перевірки і до кінця циклу. Якщо цикл закінчено, то починає виконуватися оператор, що стоїть після **Next**. Використання **Exit For** буває необхідним при достроковому виході з циклу.

Приклад 7.1

Завдання: написати програму обчислення факторіала y = **n!** (схема алгоритму подається на рис. 7.1, вигляд екранної форми на рис. 7.2).

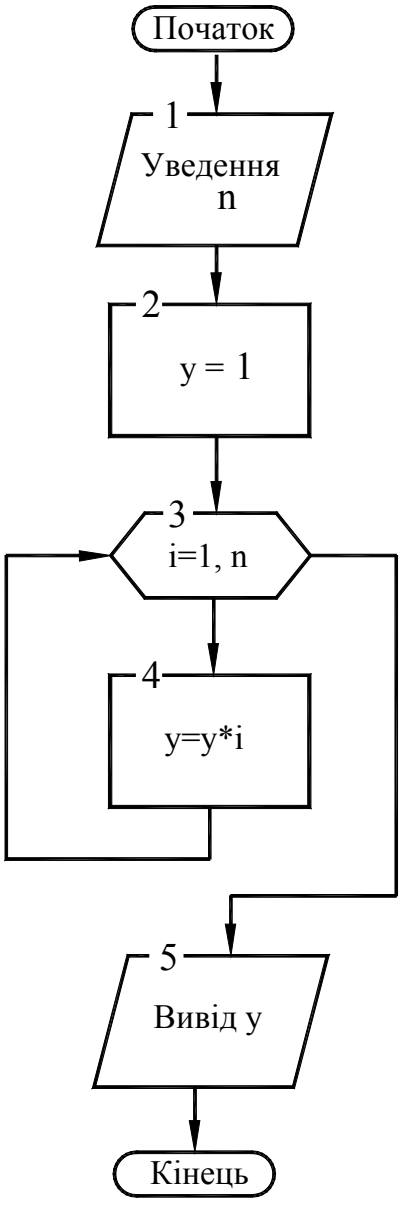


Рис. 7.1. Схема алгоритму обчислення функції **y = n!**

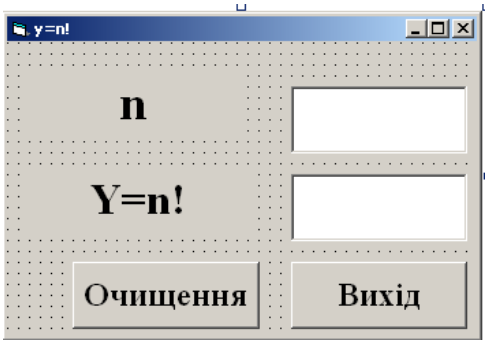


Рис. 7.2. Загальний Вигляд екранної форми для завдання з прикладу 7.1

Виконання. Для розвязування прикладу потрібно виконати операції в такій послідовності:

1. Створити електронну форму й завдати властивості її елементів, у порядку, передбаченому попередніми прикладами. При цьому текстовим полям присвоюються імена **n** та **y**, а полю для написання повідомленнь – **Mes**, імена цих кнопок – **Очищення** й **Вихід**.

2. Завдати функції кнопці **Вихід**, для чого встановити об'єкт **Вихід** і для події **Click** записати, що

```
Private Sub Вихід_C1ick()
  End
End Sub.
```

3. Завдати функції кнопці **Очищення**, встановивши об'єкт **Очищення**, а для події **Click** записати такі кроки:

```
Private Sub Очищення_Click()
n.Text = " "
y.Text = " "
Mes.caption = " "
End Sub
```

4. Виконати контроль уведення даних у поле **n** і розрахувати факторіал.

Установити в текстовому полі **n** подію **KeyPress**, далі ввести текст підпрограми:

```
Sub n_KeyPress (KeyAscii As Integer)
  Dim i As Integer ' оголошення параметра циклу як цілого
y = 1 'початкове значення y
' Контроль уведення даних
 Select Case KeyAscii
   Case 0,8,46,48 To 57
   Case 13
     For i = 1 To Val(n.Text)
       y.Text = Val(y.Text) * i
     Next i
   Case Else
     Mes.Caption = "Уведіть додатне число "
 End Select
End Sub
```

5. Зберегти проект під іменем **"Факторіал"**, таким чином:

Меню **File** → команда **Save Project** → задати ім'я файлу й папку.

6. Налагодити програму в такій послідовності операцій:

Меню Run → команда **Start**, увести такі вхідні дані контрольного прикладу:

● **n = 5** і натиснути клавішу **Enter**, має вийти така відповідь: **y** = 120;

● потім клацнути по кнопці **Очищення** й увести, що **n = –7** , при цьому з'явиться відповідне повідомлення.

7. Налагоджену програму зберегти таким чином:

Меню **File** → команда **Save Project.**

У описаному прикладі використовувалися прості змінні, що вводились у текстові вікна. У прикладі 7.2 буде розглянуто арифметичний цикл з індексованими змінними, тобто роботу з масивом даних.

7.3. Ітераційні цикли

Ітераційні цикли – це цикли, число повторень у яких заздалегідь не відоме, але є певні умови, за яких цикл повинен бути завершений. Для запису таких циклів використовуються такі оператори:

1. **Do While** умова
   операторі
Loop

Названі оператори будуть діяти доти, поки умова буде виконуватися тобто буде **правдивою**, в іншому випадку керування буде передано операторові, що стоїть після **Loop**.

2. **Do Until** умова
   операторі
Loop

У цьому випадку оператори будуть виконуватися доти, поки умова буде **помилковою**. Умова перевіряється перед кожним проходженням циклу.

Існують модифікації розглянутих операторів, коли умова переноситься в кінець циклу, після **Loop**, тоді цикл виконується перший раз обов'язково.

7.4. Складні цикли, використання меню

Складні цикли – являють собою цикли з розгалуженнями або вкладеними циклами. Для їхнього написання використовуються ті самі оператори, що й для простих циклів і розгалужених процесів.

Приклад 7.2

Завдання: створити форму (рис. 7.3) для розв'язування задачі на табулювання функції. Побудувати у цій формі головне меню, передбачивши такі команди: ***закінчити роботу програми***, ***табулювати функцію***, ***очистити поля виведення результатів***. Результати табулювання вивести у багаторядкове поле редагування (об'єкт типу **Text**).

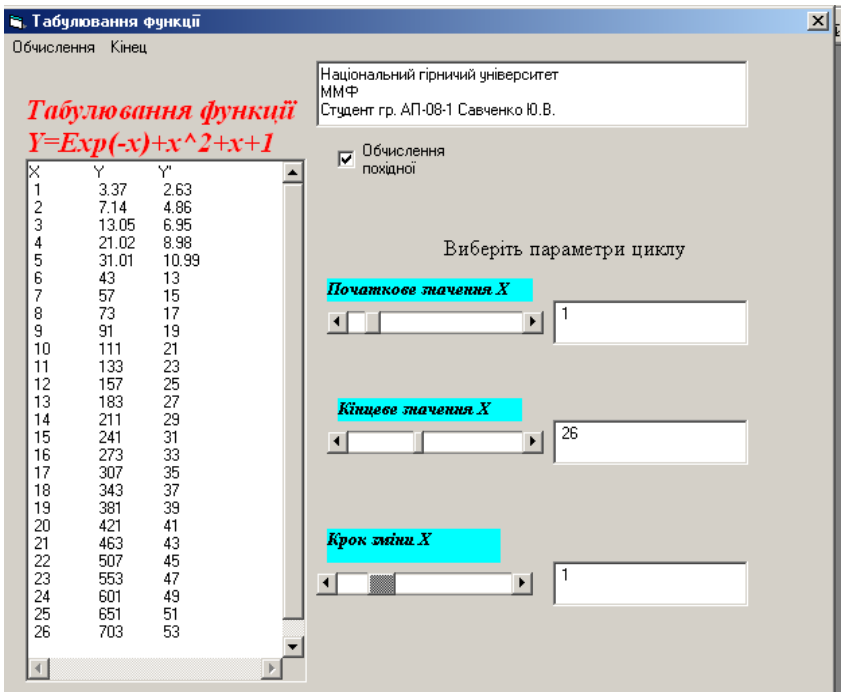


Рис. 7.3. Загальний вигляд форми для табулювання функції (приклад 7.2)

Об'єкт **CheckBox** використовують, якщо потрібно вивести результати табулювання похідної функції, що задана.

Виконання.

Для розвязування прикладу потрібно здійснити такі операції:

1. Завантажити середовище Visual Basic.

2. Змінити заголовок (**Caption**) форми "**Forml**" на "**Табулювання функції**" (без лапок) і збільшити розміри форми, а також змінити ім'я форми "**Forml**" на **Таблр**.

3. Зберегти виконану на даний момент форму у власній папці (**File → Save Project**).

4. Вставити у форму (рис. 7.4) поля редагування объект  **TextBox (Text1)**. Збільшити розміри поля. Передбачити властивість **ScrollBars** (наявність смуг прокручування) цього об'єкта, задавши оператор **Both** (будуть діяти обидві смуги – вертикальна й горизонтальна). Для властивості **MultiLine** задати оператор **True**, який дозволяє об'єкту працювати більш ніж з одним рядком тексту.

5. Розташувати у формі поля редагування вікна **Text2**, **Text3**, **Text4,** і відповідні їм текстові поля "**Початкове значення X**", "**Кінцеве значання X**", "**Крок зміни X**", а також поле для такого вигляду даної функції: **Y = Exp(–X) + X^2 + X + 1 (Label1)**. Розмір, стиль і колір шрифтів виберати на власний розсуд таким чином, щоб форма мала привабливий вигляд. Розмістити текстове поле **Text5**, призначене для подання інформації про навчальний заклад, факультет, групу і прізвище студента, який виконує завдання.

6. Задати початкові значення для полів редагування лівої і правої меж аргумента функції та для кроку зміни цього аргумента, відповідно до зразка, зображеного рис. 7.4. Для цього змінити властивість **Text** цих об’єктів.

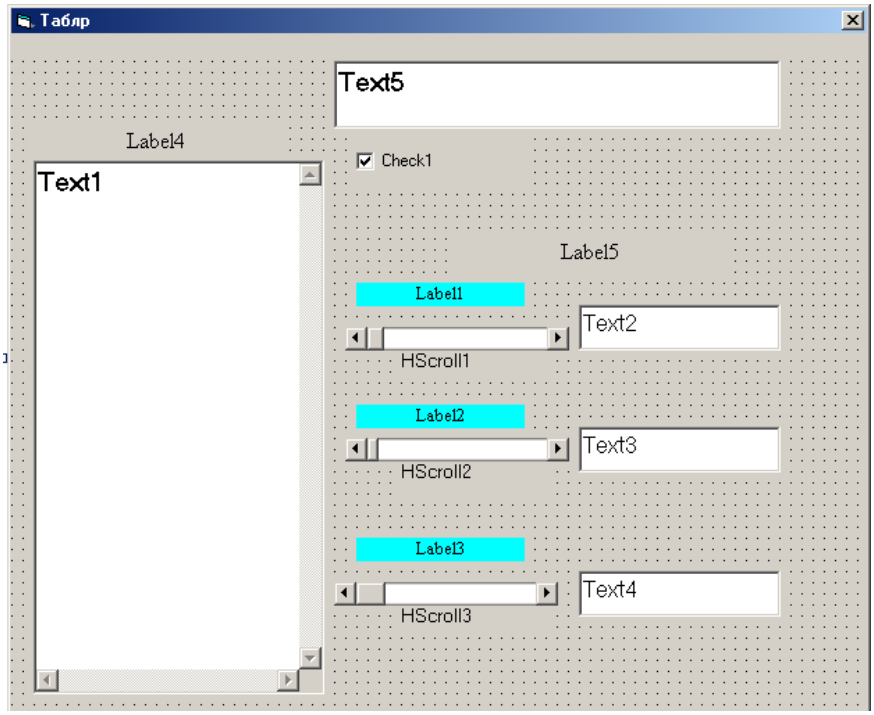


Рис. 7.4. Загальний вигляд початкових даних для створення форми розв’язку прикладу 7.2

7. Вирівняти вставлені поля редагування по лівому краю першого об'єкта та зробити їхні розміри однаковими.

8. Вставити у форму прапорець  , для чого використати компоненту **CheckBox** із палітри компонентів.

9. Ввести назви команд головного меню форми (див. рис. 7.5).

При цьому команди головного меню, як і інші компоненти Visual Basic, являють собою об'єкти. Отже для створення команд треба вибрати елемент головного меню **Tools → Menu Editor**. У вікні, що при цьому з'явиться, ввести назви команд меню (властивість **Caption**) та їхні імена (властивість **Name**), щоразу натискаючи на кнопку "**Next**" (табл. 7.1).

Таблиця 7.1

Назви пунктів головного меню і відповідні їм імена

Caption	**Name**
Обчислення	**mnuCalc**
Табулювання	**mnuTablr**
Очистити	**mnuClear**
Кінець	**mnuFinish**
Вихід	**mnuEnd**

Далі потрібно задати ієрархію команд за допомогою стрілок:  (підпорядкувати) та  (вивести із підпорядкування). Змінити послідовність команд можна за допомогою стрілок:  та  .

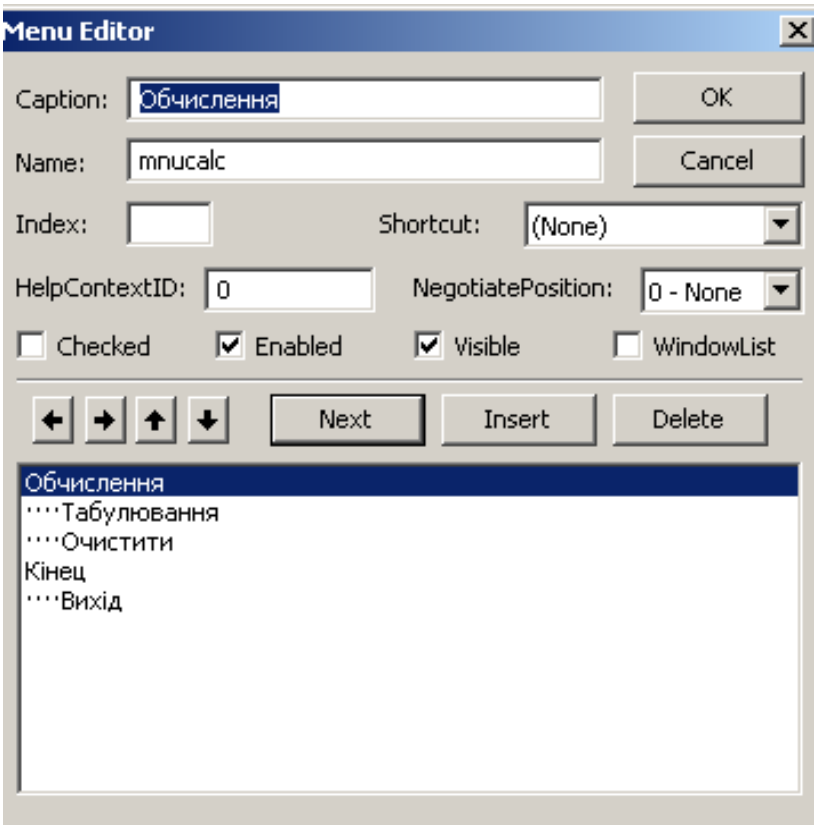


Рис. 7.5. Приклад створення головного меню в середовищі редактора **Menu Editor**

10. Запрограмувати команду "**Очистити**" головного меню. Для цього клацнути у місці команди головного меню форми "**Очистити**", не запускаючи програму на виконання, тоді з'явиться заготівка процедури реакції на подію виклику цієї команди, у ній потрібно записати команду присвоєння порожнього рядка для очистки поля редагування **Textl, Text2, Text3, Text4**, а саме:

```
Private Sub mnuClear_Click()
    Text1.Text = ""
    Text2.Text = ""
    Text3.Text = ""
    Text4.Text = ""
End Sub
```

11. Запрограмувати команду "**Кінець**" головного меню, скориставшись стандартною процедурою **End**:

```
Private Sub mnuEnd_Click()
   End
End Sub
```

12. Запрограмувати команду "**Табулювання**". Для цього клацнути один раз в її місці, тоді з'явиться заготівка процедури, яку необхідно заповнити.

Весь програмний код приведений нижче:

```
' Процедура очищення текстових вікон виведення результатів
Private Sub mnuclear_Click()
  Text1.Text = ""
  Text2.Text = ""
  Text3.Text = ""
  Text4.Text = ""
End Sub
' Процедура створення смуги прокручування початкового значення X
Private Sub HScroll1_Scroll()
  HScroll1_Change
End Sub
Private Sub HScroll1_Change()
  lb1 = Str(HScroll1)
  ' Виведення початкового значення X у вікні Text3
  ' за допомогою смуги прокручування
  Text2.Text = lb1
End Sub
' Процедура створення смуги прокручування кінцевого значення X
Private Sub HScroll2_Scroll()
  HScroll2_Change
End Sub
Private Sub HScroll2_Change()
  lb2 = Str(HScroll2)
 ' Виведення кінцевого значення X у вікні Text3
 ' за допомогою смуги прокручування
  Text3.Text = lb2
End Sub
'створення смуги прокручування кроку зміни X
Private Sub HScroll3_Scroll()
  HScroll3_Change
End Sub
Private Sub HScroll3_Change()
  lb3 = Str(HScroll3)
  ' Виведення кроку зміни X у вікні Text3
  'за допомогою смуги прокручування
  Text4.Text = lb3
End Sub
' Процедура виходу з програми за допомогою меню
Private Sub mnuend_Click() 'вихід з програми
  End
End Sub
' Процедура створення меню для роботи програми
```

```
Private Sub mnutablr_Click()
  Dim x, y As Double 'оголошення типу даних Double
  Dim NewLine, Space As String 'оголошення типу даних String
  NewLine = Chr(13) + Chr(10) 'Символ "Enter"
  Space = Chr(9)      'Символ "Tab"
  If Check1.Value = Checked Then
    ' Побудова шапки таблиці для обчислення функції Y та її похідної Y'
    Text1.Text = "X" + Space + "Y" + Space + "Y'" + NewLine
  Else
  ' Побудова шапки таблиці для обчислення функції Y
   Text1.Text = "X" + Space + "Y" + NewLine
  End If
  For x = Val(Text2.Text) To Val(Text3.Text) Step Val(Text4.Text)
 y = Exp(-x) + x ^ 2 + x + 1 ' обчислення функції Y
 Y1 = -Exp(-x) + 2 * x + 1 ' обчислення похідної Y'
     If Check1.Value = Checked Then
' Вивідення в текстовому вікні Text1
' результатів обчислення функції Y та похідної Y'
        Text1.Text = Text1.Text + Str(Format$(x, "0.00")) + Space + _
        Str(Format$(y, "0.00")) + Space + Str(Format$(Y1, "0.00")) + _
        NewLine
      Else
' Вивідення в текстовому вікні Text1 результатів обчислення функції Y
        Text1.Text = Text1.Text + Str(x) + Space + _
         Str(Format$(y, "0.00")) + NewLine
      End If
  Next x
End Sub
```

Контрольні питання

1. Яка різниця існує в написанні програм для арифметичних та ітераційних циклів?

2. Як працює оператор для запису арифметичних циклів?

3. Назвіть оператори для запису ітераційних циклів, чим вони відрізняються один від одгного?

4. Що нового з'являється в програмах зі складними циклами в порівнянні із програмами простих циклічних процесів?

5. Яким чином можна додати в проект нові форми і для чого їх можна використовувати?

6. Які можливості створює використання меню в програмі?

7. Яким чином можна використовувати в різних підпрограмах однакові змінні?

8. Напишіть проект для обчислення такої функції: y = n!, вибираючи значення **n** за допомогою лінійки прокручування.

9. Складіть програму обчислення функції $F = S^2$, де i = l–4–n, виведіть значення i, i^2, F.

10. Напишіть проект завантаження масиву чисел і вибору з нього максимального або мінімального значення (використовуючи селекторні кнопки) та вибору можливості розрахунку середнього значення введених чисел (застосовуючи прапорець).

8. ГРАФІКА В VISUAL BASIC

8.1. Загальні положення

Visual Basic дозволяє створювати програми, які працюють з графікою. Програма може вивести зображення на поверхню форми або її компоненти **PictureBox**. Для того, щоб під час роботи програми на поверхні об'єкта з'явилася, наприклад, ілюстрація або лінія, необхідно використати відповідний метод.

Графіку на поверхні об'єкта повинна формувати процедура обробки події **Paint**. Це пояснюється необхідністю оновлення графіки при кожній появі об'єкта на екрані, а подія **Paint** якраз і виникає щоразу, коли об'єкт з'являється на екрані (в т. ч. і після того, як користувач зсуне інше вікно, яке частково або повністю перекриває вікно програми).

8.2. Поняття про координатну систему

Коли виконується робота з графічними елементами або використовуються інструменти малювання, необхідно описувати, де саме на формі (або на компоненті **PictureBox**) розташовуватиметься потрібний елемент (або намалюється вибрана геометрична фігура). Щоб визначити положення того або іншого графічного елемента або образу, використовуються *координати*.

Будь-яка точка на формі або на малюнку може бути описана за допомогою пари чисел **X** і **Y**, які задають точне її розташування (**X** – горизонтальна координата, **Y** – вертикальна). На відміну від математичних позначень, у середовищі Visual Basic вертикальна координата **Y** зростає не знизу вгору, а навпаки – згори вниз, а горизонтальна координата **X** збільшується стандартно – зліва направо (рис. 8.1).

При цьому в середовищі Visual Basic робота з графічними елементами може виконуватися в різних системах координат, залежно від яких змінюється здатність розрізнювати зображення.

Найбільш звичною для користувачів, як правило, є одиниця вимірювання піксель. У цих одиницях вимірюється розрізнювальна здатність монітора. Крім того, розмір растрових малюнків також вимірюється в пікселях. Проте недолік цієї одиниці вимірювання полягає в тому, що всі графічні об'єкти, які вимірюються за допомогою пікселів, виявляються залежними від встановленої на даний момент розрізнювальної здатності екрана.

Стандартна система координат в середовищі Visual Basic припускає використання одиниць вимірювання, які називаються твіпами. Один твип дорівнює 1/20 пункта або 1/1440 дюйма. Ця одиниця вимірювання є точнішою, ніж піксель. Разом з твіпами використовуються також інші одиниці вимірювання, опис яких наведено в табл. 8.1.

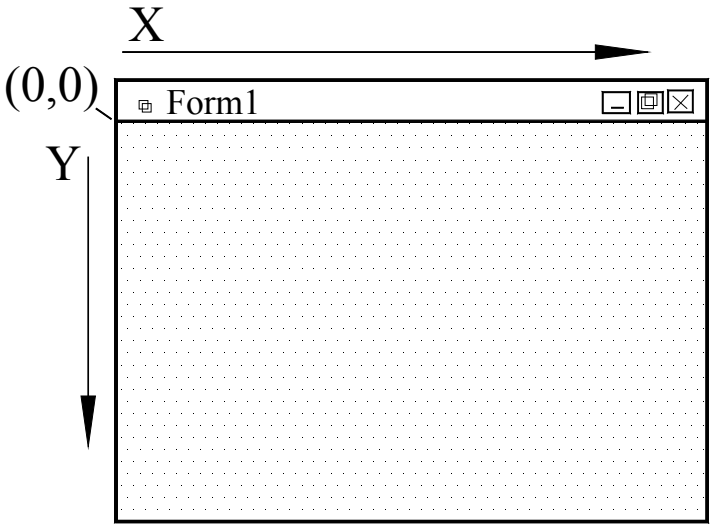


Рис. 8.1. Стандартна система координат в Visual Basic

Таблиця 8.1.
Опис одиниць вимірювання (властивість **ScaleMode**), які використовуютьсяв Visual Basic

Константа	Значення	Опис
vbCentimeters	7	Сантиметри
vbCharacters	4	Визначаються розміром символів (120x240 твіпів)
vblnches	5	Дюйми
vbMillimeters	6	Міліметри
vbPixels	3	Пікселі
vbPoints	2	Пункти (72 пункти = 1 дюйм)
vbTwips	1	Твіпи (20 твіпів = 1 пункт)
vbUser	0	Одиниці вимірювання, які призначені для користувача

З метою програмної установки користувацької системи координат використовується метод **Scale**, для якого характерний такий синтаксис:

об'єкт.Scale (XI, Yl) – (X2, Y2)

де **об'єкт** – поверхня форми **Form** або об'єкта **Picture** (компонент **PictureBox**); **X1,Y1** – координати верхнього лівого кута графічного поля в

стандартній системі координат; **X2,Y2** – координати правого нижнього кута графічного поля в теж у стандартній системі координат.

Приклад 8.1

Завдання: створити користувацьку систему координат на поверхні форми з початком відліку в лівому нижньому куті графічного поля. При цьому вісь **X** має бути спрямована вправо, а вісь **Y** – угору. Ширина графічного поля дорівнює 400 мм, а висота 300 мм (рис. 8.2).

Виконання. Фрагмент програмного коду для установки розмірів графічної зони і напрямку осей координат такий:

ScaleMode = vbMillimeters ' *опис одиниць вимірювання графічного поля*
Form1.Scale (0,300) – (400,0)

Тут **(0,300**) – координати точки 1, (**400,0**) – координати точки 2.

Для очищення поверхні графічного об'єкта (форми або об'єкта **PictureBox**) використовується метод **Cls**. Наприклад, у програмному коді для очищення форми слід увести вираз **Form1.CLS**

8.3. Позиціонування точки на графічній поверхні

У процесі малювання будь-якого зображення обов’язково існує остання точка. Ця точка називається поточною, а її координати можуть бути використані для наступної операції малювання (тоді цю дію буде виконано в так званих відносних координатах).

Наприклад, проводячі лінію в абсолютних координатах (у яких за початок відліку береться лівий верхній кут) необхідно указувати координати її початкової і кінцевої точок. Якщо ж використовувати відносні координати, де початком відліку служитиме поточна точка, то потрібно буде задати тільки параметри кінцевої точки цієї лінії.

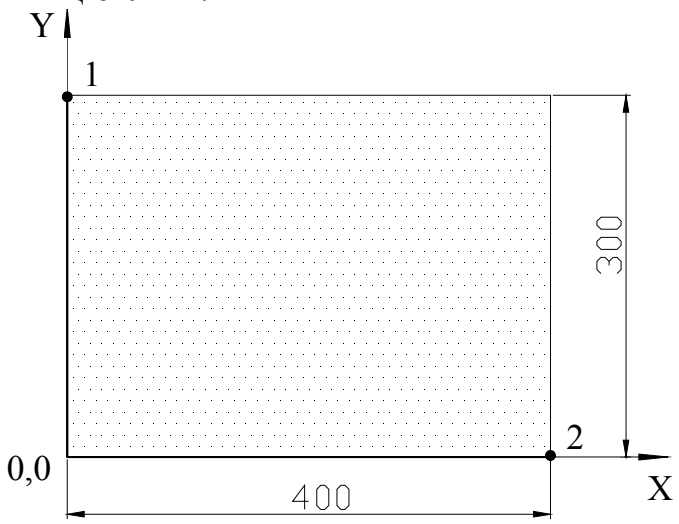


Рис. 8.2. Користувацька система координат на поверхні форми з початком відліку в лівому нижньому кутку графічного поля

Координати поточної точки форми або елемента можна встановити (або отримати), скориставшись властивостями **Currentx** і **Currenty**. Наприклад,

подані нижче команди встановлять координати (50, 100) для поточної точки форми.

Forml.CurrentX = 50

Forml.CurrentY =100

Встановити координати для поточної точки можна також накресливши точку-примітив на поверхні форми: **Pset (X,Y),** де **X,Y** – координати зображеної точки. Щоб точка-примітив не відображалась на екрані, необхідно їй присвоїти колір поверхні графічної області.

8.4. Графічні примітиви

Будь-яка картинка, креслення, схема є сукупністю графічних примітивів: точок, ліній, кіл, дуг, тексту та ін.

Викреслювання (виведення) графічних примітивів на графічній поверхні виконують за допомогою відповідних методів (табл. 8.2).

Таблиця 8.2

Методи викреслювання (виведення) графічних примітивів

Метод	**Дія**
PSet	Зображує точку
Line	Залежно від значення параметрів креслить лінію, прямокутник або контур прямокутника
Circle	Залежно від значення параметрів креслить коло, еліпс, дугу, круг або сектор
Print	Виводить текст

8.4.1. Зображення точки

Точку на графічній поверхні зображують за допомогою методу **PSet**. Інструкцію виклику методу **PSet** у загальному вигляді можна записати таким чином:

Об'єкт.Pset Step(x,y),Color

Параметр **Об'єкт** задає об'єкт, на поверхні якого треба зобразити точку. Якщо її проставляють на поверхні поточної форми, то цей параметр можна не використовувати.

Параметри **x** і **y** задають координати точки на графічній поверхні, колір якої треба змінити.

Параметр **Color** задає колір точки. У ролі цього параметра можна використовувати одну з пойменованих констант (див. табл. 8.3), а також значення функції **Rgb**, яка повертає код кольору, заданий червоною, зеленою і синьою складовими (як відомо, будь-який колір можна отримати шляхом змішування в різних пропорціях червоної, зеленої та синьої фарб). У функції **Rgb** три параметри: перший задає частку червоної (**red**), другий – зеленої (**green**) третій – синьої (**blue**) складових. Значення кожного з параметрів має перебувати в діапазоні від 0 до 255. Наприклад, значенням функції **Rgb** (205, 127, 50) є код "золотого" кольору. Параметр **Color** не є обов'язковим. Якщо

його не названо, то колір зафарбовування точки залежить від значення властивості **Forecolor** графічної поверхні, для якої застосовано метод.

Таблиця. 8.3

Константи, які використовуються при завданні кольору

Константа	**Колір**
vbBlack	Чорний
vbRed	Червоний
vbGreen	Зелений
vbYellow	Жовтий
vbBlue	Синій
vbMagenta	Пурпурний
vbCyan	Бірюзовий
vbWhite	Білий

Ключове слово **Step** можна не зазаначати. У цьому випадку параметри **x** і **y** задають абсолютні координати точки. Якщо слово **Step** вживається, то координати точки роблять свій відлік від поточного положення покажчика графічного виведення.

Розмір (діаметр) точки, яку зображують з використанням методу **Pset**, залежить від поточного значення властивості **Drawwidth** поверхні.

8.4.2. Проведення лінії

Викреслювання прямої лінії відбувається за допомогою методу **Line**.

Інструкція виклику методу **Line**, що забезпечує викреслювання лінії, в загальному вигляді записується таким чином:

Об'єкт.Line Step(x1,y1) – Step(x2,y2), Color

Параметр **Об'єкт** задає об'єкт, на поверхні якого треба провести лінію. Якщо це відбувається на поверхні форми, то цей параметр можна не використовувати.

Параметри **x1** і **y1** задають координати початкової точки лінії, а параметри **x2** і **y2** – координати точки кінця.

Параметр **Color** задає колір лінії. У ролі цього параметра можна використовувати одну з пойменованих констант (див. табл. 8.3) або значення функції **Rgb**. Параметр **Color** не вважається обов'язковим. Якщо його не вжито, то колір лінії залежить від значення властивості **Forecolor** графічної поверхні, на якій використано метод **Line**.

При цьому ключове слово **Step** можна не вживати. Тоді параметри **x1**, **y1** і **x2**, **y2** задають абсолютні координати кінця лінії. Якщо слово **Step** вжито перед параметрами **x1**, **y2**, координати початкової точки лінії відліковуються від

покажчика графічного виведення. Коли ж слово **Step** вжито перед параметрами **x2**, **y2**, то координати кінцевої точки лінії починають свій відлік від її початку.

Товщину і вид (стиль) лінії визначають відповідно до властивостей **DrawWidth** і **DrawStyle** графічної поверхні, на якій використовують метод **Line**. У табл. 8.4 перелічено константи, використовуючи які, можна задати вид лінії. Слід звернути увагу: лінія, товщина якої більш ніж 1 піксел, може бути тільки суцільною, бо провести її пунктирною неможливо.

Таблиця 8.4

Константи, для використовуються для завдання графічної поверхні лінії

Константа	**Вид (стиль) лінії**
VbSoli	Суцільна
VbDash	Штрихова (довгі штрихи)
VbDot	Пунктирна (короткі штрихи)
VbDashdot	Штрих-пунктирна
VbDashDotDot	Штрих, два пунктири

8.4.3. Креслення прямокутника

Метод **Line** дозволяє зобразити не тільки лінію, але й прямокутник (див. рис. 8.3).

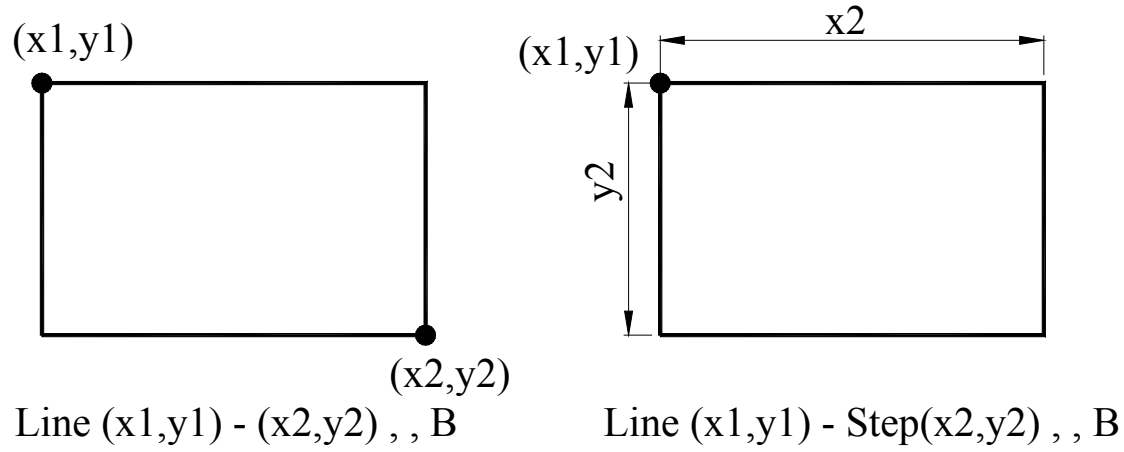


Рис. 8.3. Використання методу **Line** для креслення прямокутника

Інструкцію виклику методу **Line**, що забезпечує зображення прямокутника, в загальному вигляді записуємо таким чином:

Об'єкт.Line Step(x1,y1) – Step(x2,y2),Color,B

Параметр **Об'єкт** задає об'єкт, на поверхні якого треба накреслити прямокутник. Якщо це відбувається на поверхні поточної форми, то параметром **Об'єкт** можна не користуватись.

Параметри **x1** і **y1** задають координати лівого верхнього (нижнього) кута прямокутника, а параметри **x2**, **y2** – координати правого нижнього (верхнього) кута.

Параметр **B** вказує на те, що за допомогою методу **Line** потрібно накреслити прямокутник, його колір задає параметр **Color**. У ролі цього параметра можна використовувати одну з пойменованих констант (див. табл. 8.3) або значення функції **Rgb**. Параметр **Color** вважається обов'язковим. Якщо його не використано, то колір залежить від значення властивості **ForeColor** графічної поверхні, на якій застосовано метод.

Товщину і вид (стиль) лінії прямокутника визначають відповідно до властивостей **DrawWidth** і **DrawStyle** графічної поверхні, на якій використано метод **Line**. У ролі властивості **DrawStyle** можна використовувати одну з наведених у табл. 8.4 констант. Якщо товщина лінії, яка обмежує прямокутник більш ніж 1 піксель, то розмір прямокутника (по зовнішній межі) буде більшим від розміру поля, заданого параметрами **x1**, **y1** і **x2**, **y2**. Щоб розмір прямокутника (по зовнішній межі) дорівнював розміру графічного поля (у тому випадку, коли ширина межі перевищує 1 піксель), то властивості **DrawStyle** треба присвоїти значення **vbInsideSoiid**. Тоді лінія межі буде накреслена так, що її зовнішній край буде розташований точно в межах графічного поля, заданого точками (**x1, y1**) і (**x2, y2**).

Колір і стиль зафарбовування внутрішнього поля прямокутника визначають відповідно до властивостей **FillColor** і **FilStyle** тієї графічної поверхні, на якій застосовано метод. За умовчуванням значення властивості **FillStyle** відповідає **vbSFTransparent**, тому метод застосовується тільки для зображення меж прямокутника. Щоб внутрішнє поле прямокутника було зафарбовано, задане властивістю **FillColor** значення властивості **FillStyle** має відповідати **vbFSSoiid** (суцільне тонування). Внутрішнє поле прямокутника також може бути заштриховане. Константи, за допомогою яких можна задати стиль зафарбовування, наведені в табл. 8.5.

Ключове слово **Step** можна не вживати. В цьому випадку параметри **x1, y1** і **x2, y2** задають абсолютні координати кутів прямокутника.

Якщо слово **Step** вживається перед параметрами **x1, y1**, то координати лівого верхнього (нижнего) кута прямокутника починають відлік від поточного положення покажчика графічного виведення.

Якщо слово **Step** зазначається перед параметрами **x2, y2**, то координати правого нижнього (верхнього) кута прямокутника починають відлік від іншого діагонального кута, тобто фактично ці параметри задають розмір прямокутника (**x2** – ширина, **y2** – висота).

Якщо замість параметра **B** вжито параметр **BF**, то метод передбачає зображення зафарбованого прямокутника, колір межової лінії та колір зафарбовування якого збігатимуться. Колір прямокутника в цьому випадку визначить параметр **Color** або (якщо його не використовують) властивість **ForeColor**. Слід звернути увагу, що після того як за допомогою методу **Line** буде намальовано прямокутник, покажчик графічного виведення перебуватиме в точці (**x2, y2**).

Таблиця 8.5
Стилі зафарбовування внутрішнього поля прямокутника

Константа	Спосіб (стиль) зафарбовування
vbFSTranspparent	Внутрішнє поле прямокутника не фарбується
vbFSSolid	Звичайне (суцільне) зафарбовування
vbHorisontalLine	Горизонтальне штрихування
vbVerticalLine	Вертикальне штрихування
vbUpwardDiagonal	Діагональне штрихування (нахил ліній уліво)
vbDownwardDiagonal	Діагональне штрихування (нахил ліній управо)
vbCross	Клітинка
vbDiagonalCross	Діагональна клітинка

8.4.4. Зображення кола й круга

Метод **Circle** залежно від значення параметрів дозволяє креслити коло, еліпс, дугу або сектор. Інструкцію виклику методу **Circle**, що забезпечує зображення кола або круга в загальному вигляді можна записати таким чином:

Об'єкт.Circle Step(x,y), r, Color

Параметр **Об'єкт** задає об'єкт, на поверхні якого треба накреслити коло. Коли це відбувається на поверхні форми, то цим параметром можна не користуватись.

Параметри **x1** і **y1** задають координати центра кола (круга). Якщо вжито слово **Step**, то положення центра починає відлік від поточного положення покажчика графічного виведення.

Параметр **r** задає значення радіуса кола (круга).

Параметр **Color** визначає колір кола або межової лінії. У ролі цього параметра можна використовувати одну з пойменованих констант (див. табл. 8.3) або значення функції **Rgb**. Параметр **Color** не вважається обов'язковим. Якщо цей параметр не використано, то колір кола або межової лінії залежить від значення властивості **ForeColor** графічної поверхні, на якій застосовано метод.

Товщину і вид (стиль) лінії кола або межової лінії визначають відповідно до властивостей **DrawWidth** і **DrawStyle** графічної поверхні, на якій використано метод.

Колір і стиль зафарбовування внутрішнього поля кола визначають відповідно до властивостей **FillColor** і **FillStyle** тієї графічної поверхні, на якій

застосовано метод. Щоб внутрішню область кола було зафарбовано, значення властивості **FillStyle** має відрізнятись від значення **vbFSTransparent**.

8.4.5. Креслення дуги й сектора

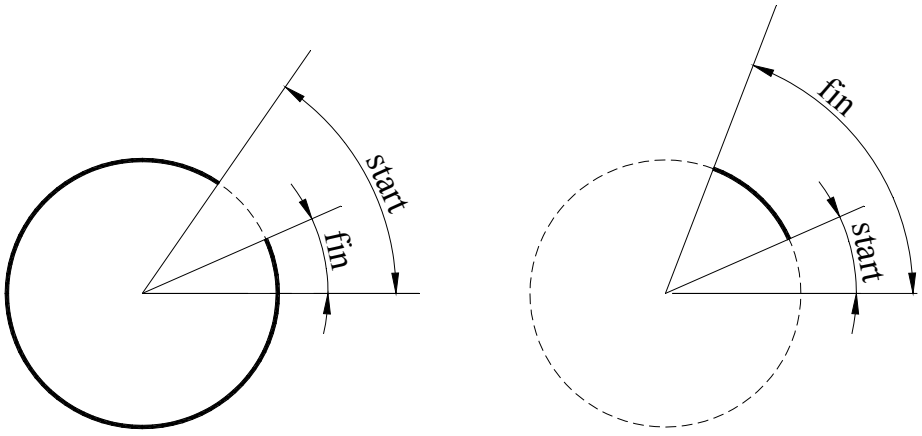
Інструкцію виклику методу **Circle**, що забезпечує зображення дуги або сектора в загальному вигляді, можна записати таким чином:

Об'єкт.Circle Step(x,y),r, Color, start, fin

За допомогою параметрів **x, y, r** і **Color** визначають відповідно координату центра кола, з якого вирізана дуга (сектор), радіус кола і колір лінії дуги (межі сектора). Товщину і стиль лінії дуги або межі сектора, а також колір і стиль тонування внутрішнього поля сектора визначають відповідно, використовуючи параметри **DrawWidth**, **DrawStyle**, **FillColor** і **FillStyle** графічної поверхні, на якій застосовується метод **Circle**.

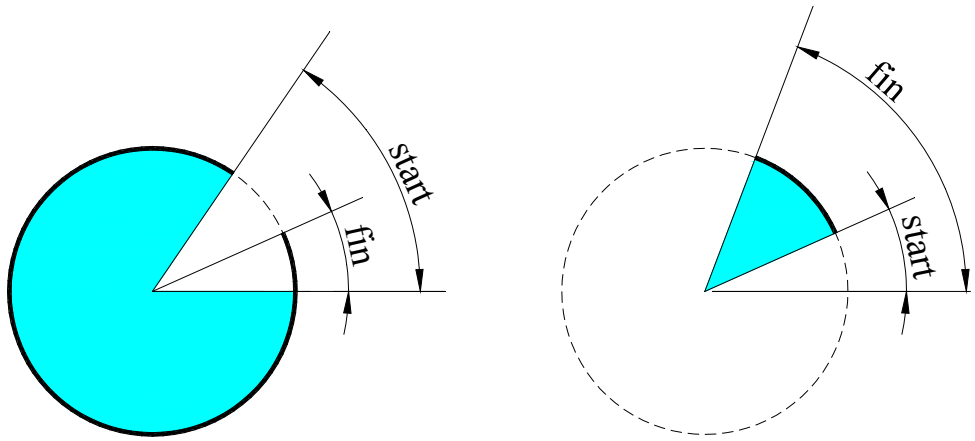
Параметр **Start** задає початкову точку дуги, яка лежить на перетині лінії кола і прямої, проведеної з центра кола під кутом **Start** до осі **X**.

Параметр **fin** задає кінцеву точку дуги (див. рис. 8.4). Кутові координати вимірюються в радіанах і зростають у напрямку проти годинникової стрілки. Дуга викреслюється від початкової точки до кінцевої також проти годинникової стрілки. Слід звернути увагу на те, що метод **Circle** застосовується для викреслювання дуг, коли значення параметрів **start** і **fin** додатні, а якщо перед параметрами поставлено знак мінус, то викреслюється зображення сектора (рис. 8.5). Щоб зобразити сектор з точки, яка відповідає куту 0 градусів, замість нуля, треба записати: 2*3.14.



Circle (x,y), r, , start, fin

Рис. 8.4. Використання параметрів **start** і **fin** для креслення дуги



Circle (x,y), r, , start, fin

Рис. 8.5. Зображення сектора за допомогою методу Circle

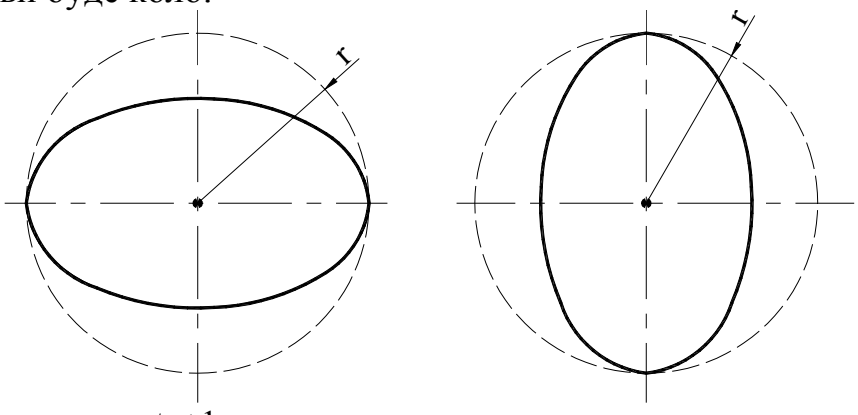
8.4.6. Зображення еліпса

Метод **Circle** дозволяє також зображувати також еліпс, еліптичну дугу або еліптичний сектор. Інструкцію виклику методу **Circle**, який забезпечує викреслювання еліпса, в загальному вигляді можна записати таким чином:

Об'єкт.Circle Step(x,y), r, Color, start, fin, aspect

Уводячи параметри **x** і **y** визначають координату центра еліпса, параметр **Color** – встановлює колір межової лінії еліпса. Товщину і стиль лінії еліпса, дуги або сектора, а також колір і стиль тонування внутрішнього поля еліпса (сектора) визначають відповідно, користуючись параметрами **DrawWidth**, **DrawStyle**, **FillColor** і **FillStyle** графічної поверхні, на якій застосовано метод **Circle**.

Користуючись параметрами **start** і **fin,** задають точки початку і кінця еліптичної дуги. При цьому параметр **r** відповідає більшому радіус еліпса, а параметр **aspect** – коефіцієнту стискування (трансформації) радіуса. Якщо значення параметра **aspect** менше від одиниці, то еліпс будується шляхом стискування кола по вертикалі, а якщо більше, то по горизонталі (рис. 8.6). У разі якщо значення параметра **aspect** дорівнює одиниці, то результатом побудови буде коло.



Circle (x,y), r,,,,aspect

Рис. 8.6. Креслення еліпса за допомогою методу **Circle**

8.4.7. Відображення тексту

Виведення тексту на графічну поверхню відбувається методом **Print**. Інструкцію виклику методу в загальному вигляді можна записати таким чином:

Об'єкт.Print *Рядок*

Параметр ***Рядок*** задає розмір рядка, який треба вивести. Позиція, в якій з'явиться рядок, залежить від поточного положення покажчика графічного виведення.

Перш ніж викликати метод **Print**, треба встановити покажчик графічного виведення (присвоїти значення властивостям **CurrentX** і **CurrentY**) в ту точку форми, де має бути лівий верхній кут текстового поля. Наприклад:

```
Private Sub Form_Click()
  Scale (0, 0) – (100, 100) 'завдання розмірів графічного поля і напрямку
  ‘координатних осей
 ' Координати початку тексту
  CURRENTX = 10 ' координата
  CURRENTY = 10
  Print "Національний гірничий університет"
End Sub
```

Результат виконання цих інструкцій бачимо на рис. 8.7.

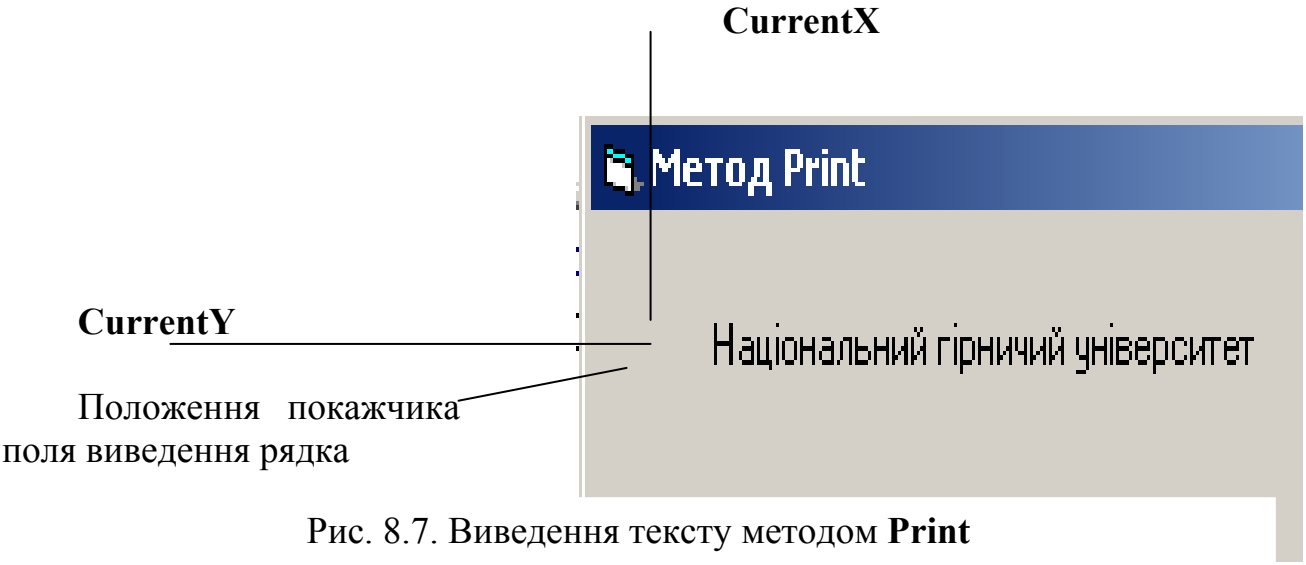


Рис. 8.7. Виведення тексту методом **Print**

Після виведення рядка покажчик автоматично переміщається в точку (**0,Y**). При цьому значення координати **Y** залежить від висоти текстового поля, яка, у свою чергу, залежить від характеристики (розміром) шрифту, що при цьому використовується для виведення тексту.

Характеристику шрифту, який застосовується в методі **Print** для відображення тексту, впливає на властивість **Font** графічної поверхні, колір символів, тобто **ForeColor**.

Якщо в ролі параметра методу **Print** увести змінну або вираз, тип якого відрізняється від параметра **String**, то відбудується автоматичне перетворення значення в рядок. Наприклад, у програмі оголошено змінну **Today** типу **Date.** Далі виконуємо такі інструкції:

```
Today = Now  ' значення функції Now – поточна дата і час
Currentx =10
CURRENTY =10
Print Today
```

Це означає, що у форму буде введено поточну дату і час.

Найбільш часто використовувані формати рядків розглянуто в табл. 8.6

Таблиця 8.6

Формати відображення даних

Формат	Опис	Приклад використання
#.##	Дробове число з двома знаками після коми (десяткового роздільника). Якщо дробова частина числа дорівнює нулю, то її знаки не відображаються	**Format(x, "#.##")**
#0.00	Дробове число з двома знаками після коми (десяткового роздільника). Виконується округлення	**Format(x, "#0.00")**
### ### ##0.00	Дробове число з двома знаками після десяткового роздільника. Цифри цілої частини об'єднані в групи по три і розділені пропусками. Виконується округлення. Використовується для відображення грошових сум (у цьому випадку в кінці рядка зазвичай додають позначення грошової одиниці)	**Format(summ "### ### ##0.00" грн.")**
#0.00%	Відсоток. Значення автоматично множиться на 10 і в кінці рядка додається символ відсотка	**Format(discount, "#0.00%")**
dd/mm/yy	Дата у форматі день, місяць, рік. Символ-роздільник визначає операційна система	**Format(Now, "Сьогодні dd/mm/yy")**
ddd	День тижня	**Format(Now, "Сьогодні dd/mm/dddd")**
mmmm	Назва місяця в повному форматі	**Format(Now, "Сьогодні dd mmmm, dddd")**

Для перетворення значення виразу в рядок потрібного формату доцільно використати функцію **Format**. Вона має два параметри: вираз, значення якого треба перетворити в рядок, і рядок форматування. Наприклад, значення **Format(C, "### ### ##0.00 грн.")** являє собою подання змінної **C**, у вигляді рядка в якому цифри, об'єднано в групи по три, розділено пропусками, а після

десяткового роздільника завжди відображаються дві цифри (навіть якщо дробова частина числа дорівнює нулю), при цьому в кінеці рядка додається позначення грошової одиниці.

Слід звернути увагу на те, що форматування наведеного рядка забезпечує автоматичне округлення числа за відомими правилами. Наприклад, якщо значення змінної **C** дорівнює **28580,446**, то значенням функції **Format** буде такий рядок: **28580,45**.

Для того, щоб повернутися до поточної дати і часу потрібно використати функцію **NOW**.

8.5. Виконання ілюстрацій

Компоненти **PictureBox** та **Image** забезпечують відображення ілюстрацій у таких форматах: **BMP**, **GIF**, **JPG**, **JPEG**, **PNG**, а також метафайлів (**WMF**, **EMF**) і значків (**ICO**). Компонент **PictureBox** має ширші, порівняно з компонентом **Image**, можливості (зокрема, на його поверхні можна створювати зображення). Разом з тим компонент **PictureBox** не дозволяє відображати ілюстрацію, розмір якої його перевищує (для виконання великих ілюстрацій слід використовувати компонент **Image**, який має здатність до масштабування). Основні властивості компонента **PictureBox** описано в табл. 8.7.

Таблиця 8.7

Властивості компонента **PictureBox**

Властивість	Опис
Picture	Ілюстрація, що відображається в полі компонента
AutoSize	Ознака автоматичної зміни розміру компонента відповідно до розміру ілюстрації. Якщо значення рівне **True**, то розмір компонента відповідає розміру ілюстрації
Visible	Ознака відображення компонента на поверхні форми. Дозволяє приховати компонент (**False**) або зробити його видимим (**True**)
Appearance	Стиль компонента. Поле компоненту може перебувати над поверхнею форми (у цьому випадку значення властивості дорівнює **3D**), або розміщуватись на одному рівні з поверхнею форми (у цьому випадку властивість набуває значення **Flat**)
BorderStyle	Тип межі компонент. Межа може бути тонкою (**FixedSingle**) або повністю відсутньою (**None**)
ScaleMode	Одиниця вимірювання розмірів компонента і об'єктів на ньому
ScaleWidth	Ширина робочого поля компонента (без урахування ширини його лівої і правої меж)
ScaleHeight	Висота робочого поля компонента (без урахування ширини його нижньої і верхньої меж)
Left	Відстань від лівої межі компонента до лівої межі форми

Закінчення табл. 8.7

Top	Відстань від верхньої межі компонента до верхньої межі форми
Width	Ширина компонента з урахуванням ширини межі
Height	Висота компонента з урахуванням ширини межі

Ілюстрацію, яка відображається в полі компонента **PictureBox**, можна задати як під час розробки форми додатка, так і в поцесі роботи програми.

Щоб задати ілюстрацію під час розробки форми, треба в рядку властивості **Picture** клацнути на кнопці з трьома крапками і в діалоговому вікні **Load Picture** вибрати файл ілюстрації. Слід звернути увагу: файл, в якому зберігається ілюстрація, вибрано таким чином, що під час роботи програми він не потрібний (Visual Basic автоматично поміщає копію ілюстрації у виконуваний файл програми).

Якщо розмір ілюстрації більший від розміру компонента, але не перевищує розміру поля форми, яка може бути використана для відображення цієї ілюстрації, то властивості **Autosize** можна присвоїти значення **True**. Унаслідок цього результаті розмір компонента буде відповідати розміру вибраної **ілюстрації**. Якщо є обмеження на розмір поля, то замість компонента **PictureBox** доведеться використовувати компонент **Image**.

Основні властивості компонента **Image** перелічено в табл. 8.8.

Таблиця 8.8

Властивості компонента **Image**

Властивість	Опис
Picture	Ілюстрація, що відображається в полі компоненту
Stretch	Ознака необхідності масштабування ілюстрації так, щоб вона займала все поле відображення. Масштабування відбувається, якщо властивість набуває заначення **True**
Width	Ширина компонента (зона відображення ілюстрації) з урахуванням ширини лінії, яка обмежує ілюстрацію
Height	Висота компонента з урахуванням ширини лінії, яка обмежує ілюстрацію
Left	Відстань від лівої межі компонента до лівої межі форми
Top	Відстань від верхньої межі компонента до верхньої межі форми
Visible	Ознака відображення компонента. Дозволяє приховати компонент (тоді вона набуває значення **False**) або зробити його видимим (тоді це значення **True**)

Закінчення табл. 8.8

Appearance	Стиль компонента. Поле компонента перебуває над поверхнею форми (у цьому випадку властивість набуває значення **3D**) або розміщується на одному рівні з поверхнею форми (у цьому випадку властивість набуває значення **Flat**)
BorderStyle	Тип межі компонента. Лінія, що обмежує ілюстрацію може бути тонкою (**FixedSingle**) або зовсім відсутньою (**None**)

Слід звернути увагу, що після завантаження в компонент **Image** ілюстрації якщо властивість **Stretch** відповідає значенню **True,** розмір компонента буде відповідати розміру ілюстрації. Якщо властивість **Stretch** набуває значення **False**, то зміна значення властивості **Picture** не приведе до зміни значень властивостей **Width** і **Height**.

Завантаження ілюстрації в поле компонента **PictureBox** (**Image**) під час роботи програми забезпечує функція **LoadPicture**. У ролі параметра цієї функції треба назвати файл ілюстрації, а її значення присвоїти властивості **Picture**. Наприклад, унаслідок виконання такої інструкції: **Picture1.Picture = LoadPicture ("d:\images\ufo.bmp")**, у полі компонента **Picture1** з'явиться ілюстрація, що зберігається в названому файлі.

Приклад 8.2

Завдання: розробити програму для побудови графіка функції

$y = \sqrt{x} e^{sin(px)}$.

У програмі передбачено:

– побудувати зображення графіка функції в заданому інтервалі;

– розмістити на поверхні екранної форми мінімальні й максимальні значення змінних **X** та **Y**;

– побудувати координатні осі й масштабну сітку для графіка;

– виконати автоматичне оцифрування шкали;

– на поверхні екранної форми з правого боку виконати зображення Державного прапора України.

Виконання:

1. Створити екранну форму відповідно до рис. 8.8.

2. Послідовно виконати операції, спрямовані на створення форми "Вибір функції", перелік яких подано в табл. 8.9.

3. Отримати остаточний вигляд екранної форми зображений на рис. 8.9.

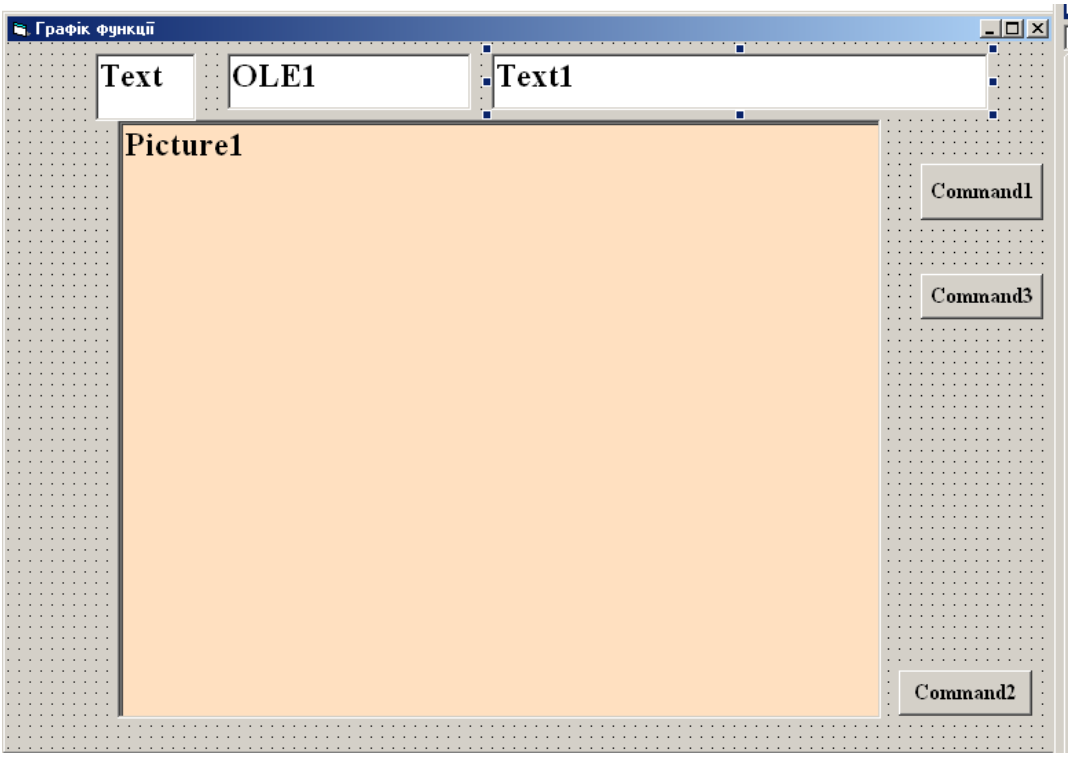


Рис. 8.8. Загальний вигляд початкових даних для створення форми виконання завдання з прикладу 8.2

Таблиця 8.9

Порядок створення екранної форми – "Графік функції"

Операція	Властивість об'єкта	Значення властивості
1. Дати ім'я електронній формі	**Name**	***Form1***
	Caption	***Графік функції***
2. Створити текстове вікно для відображення значень гальмівного шляху (інструмент **TextBox**)	**Name**	***Text1***
	Text	***Виконав студент гр. АП***
	Multline	***True***
3. Створити текстове вікно для відображення значень гальмівного шляху (інструмент **TextBox**)	**Name**	***Text2***
	Text	***Графік функції***
	Multline	***True***
4. Створити командну кнопку для виконання розрахунку гальмівного шляху (інструмент **CommandButton)**	**Name**	***Command1***
	Caption	***Запуск програми***

Закінчення табл. 8.9

5. Створити командну кнопку для виходу з програми (інструмент **CommandButton)**	**Name**	***Command3***
	Caption	***Вихід***
6. Створити командну кнопку для очищення вікон (інструмент **CommandButton)**	**Name**	***Command2***
	Caption	***Очищення***
7. Створити вікно для побудови графіка інтенсивності пасажиропотоку (інструмент **PictureBox)**	**Name**	***Picture1***
	AutoRedraw	***True***
8. Сотворити вікно для вставки виду функціональної залежності **Y** (інструмент **OLE**)	**Name**	***OLE1***

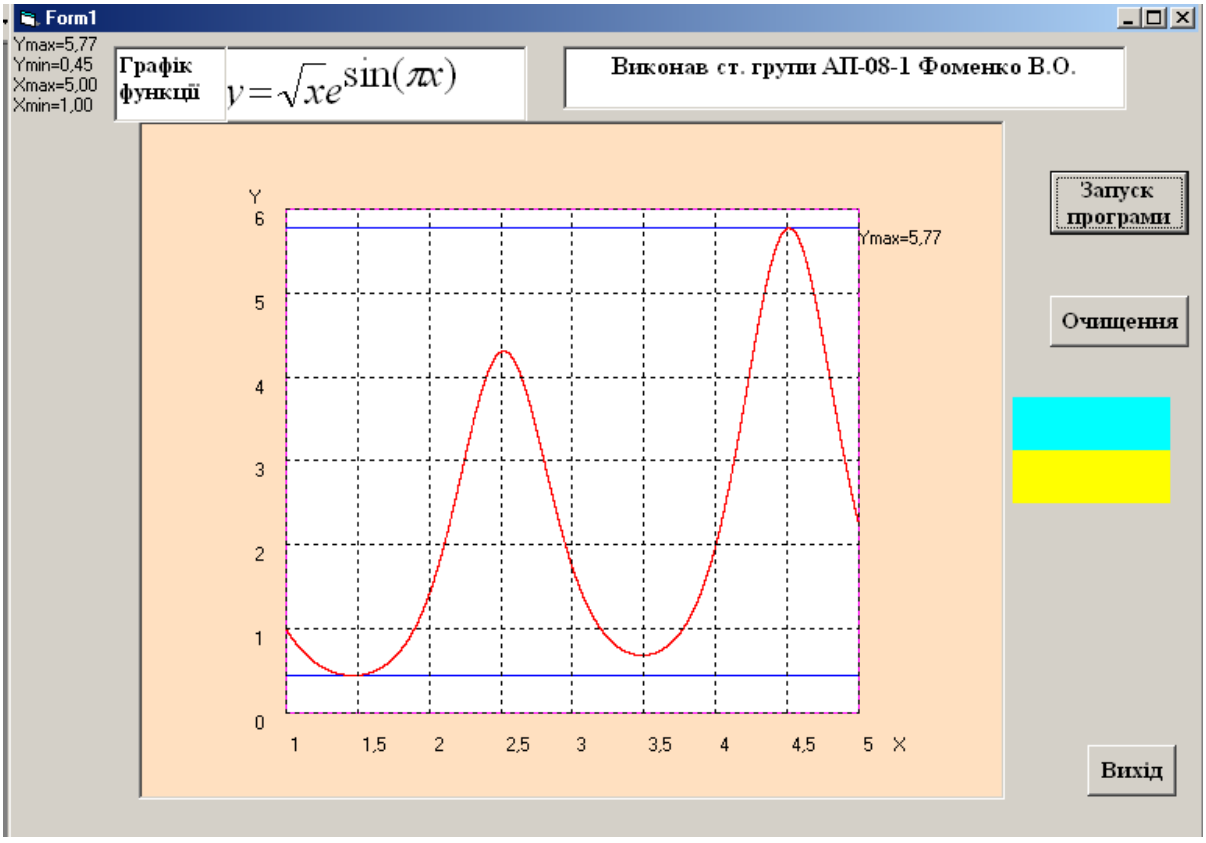


Рис. 8.9. Остаточний вигляд форми для побудови графіка функції

4. Створити програмний код побудови графіка функції таким чином:

```
' Процедура графічних побудов
Sub graf_fun()
   Dim ymin As Single
   Dim ymax As Single
```

```
ymin = –100000000#
ymax = 1000000000#
' Введення мінімальних і максимальних значень аргументу
xmin = Val(InputBox("введення XMIN="))
xmax = Val(InputBox("введення XMAX="))
' Організація циклу для перебування в інтервалі xmin <= x <= xmax
'  мінімальних і' максимальних значень функції Y
For x = xmin To xmax Step 0.001
    y = (Sqr(x)) * Exp(Sin(3.14159 * x))
    If y > ymin Then
       ymmax = y
       ymin = y
    End If
    If y < ymax Then
       ymmin = y
       ymax = y
    End If
Next x ' кінець циклу
' Друк на поверхні форми мінімальних і'максимальних значень змінних x
 ‘та y
Print "Ymax=";
Print "Ymin="; (Format(ymmin, "0.00"))
Print "Xmax="; (Format(xmax, "0.00"))
Print "Xmin="; (Format(xmin, "0.00"))
ymaxlin = Int(ymmax) + 1
yminlin = Int(ymmin)
dx = xmax – xmin
dy = ymaxlin – yminlin
' Масштабування площі вікна зображення
Picture1.Scale ((xmin - 1), (ymaxlin + 1))-((xmax + 1), (yminlin - 1))
' Зафарбування в білий колір прямокутної зони кривої графіка
Picture1.Line (xmin, yminlin)-(xmax, ymaxlin), vbWhite, BF
' Викреслювання прямокутника графічної зони побудови графіка
Picture1.Line (xmin, yminlin)-(xmax, ymaxlin), vbMagenta, B
Picture1.DrawStyle = vbSolid
' Проведення горизонтальної лінії, яка відповідає  максимальному
' значенню функції Y
Picture1.Line (xmin, ymmax)-(xmax, ymmax), vbBlue
Picture1.Print "Ymax="; (Format(ymmax, "0.00"))
' Проведення горизонтальної лінії, яка відповідає мінімальному
' значенню функції Y
Picture1.Line (xmin, ymmin)-(xmax, ymmin), vbBlue
' Організація циклу для побудови графіка
For x = xmin To xmax Step 0.0001
    y = (Sqr(x)) * Exp(Sin(3.14159 * x))
```

```
    Picture1.PSet (x, y), vbRed
Next x ' кінець циклу
Picture1.PSet (xmax, ymmax)
Picture1.PSet (xmin, ymmin)
Picture1.PSet (xmax, ymmin)
Picture1.PSet (xmin, ymmin)
Picture1.CurrentX = xmin - 0.25
Picture1.CurrentY = ymaxlin + 0.25
Picture1.Print "Y" ' позначення осі ординат Y
Print
'--------------------------------------------------------------------
' Визначення кроку масштабної сітки вздовж осі ординат Y
If dy <= 1 Then
  sty = 0.1
ElseIf dy <= 2 Then
  sty = 0.2
ElseIf dy <= 5 Then
  sty = 0.5
ElseIf dy <= 10 Then
  sty = 1
ElseIf dy <= 20 Then
  sty = 2
ElseIf dy <= 50 Then
  sty = 5
ElseIf dy <= 100 Then
  sty = 10
ElseIf dy <= 500 Then
  sty = 50
ElseIf dy <= 1000 Then
  sty = 100
ElseIf dy <= 2000 Then
  sty = 200
ElseIf dy <= 5000 Then
  sty = 500
ElseIf dy <= 10000 Then
  sty = 1000
End If
For zx = yminlin To ymaxlin Step sty
   Picture1.DrawStyle = vbDot
   Picture1.Line (xmin, zx)-(xmax, zx)
   Picture1.CurrentX = xmin - 0.25
   Picture1.CurrentY = zx
   Picture1.Print zx
Next zx
Picture1.CurrentX = xmax + 0.25
```

```
   Picture1.CurrentY = yminlin - 0.25
   Picture1.Print "X" ' позначення осі абсцис X
  Print
  ' Визначення кроку масштабної сітки вздовж осі абсцис X
    If dx <= 1 Then
        stx = 0.1
   ElseIf dx <= 2 Then
        stx = 0.2
   ElseIf dx <= 5 Then
        stx = 0.5
   ElseIf dx <= 10 Then
        stx = 1
   ElseIf dx <= 20 Then
        stx = 2
   ElseIf dx <= 50 Then
        stx = 5
   ElseIf dx <= 100 Then
        stx = 10
   ElseIf dx <= 500 Then
        stx = 50
   ElseIf dx <= 1000 Then
        stx = 100
   ElseIf dx <= 2000 Then
       stx = 200
   ElseIf dx <= 5000 Then
        stx = 500
   ElseIf dx <= 10000 Then
        stx = 1000
   End If
  ' Організація циклу для нанесення масштабної сітки
   For zy = xmin To xmax Step stx
     Picture1.Line (zy, yminlin)-(zy, ymaxlin)
     Picture1.CurrentX = zy
     Picture1.CurrentY = yminlin - 0.25
     Picture1.Print zy
   Next zy ' кінець циклу
   ' Побудова зображення Державного прапора України
   Line (9600, 3500)-(11100, 4000), vbCyan, BF
   Line (9600, 4000)-(11100, 4500), vbYellow, BF
End Sub ' Процедура запуску програми
Private Sub Command1_Click()
   Call graf_fun
End Sub ' Процедура виходу з програми
Private Sub Command2_Click()
    End
```

```
End Sub
Private Sub Command3_Click() ' Процедура очищення вікон
    Picture1.Cls
    Form1.Cls
End Sub
```

Приклад 8.3

Завдання: на основі хронометричних досліджень встановлено, що інтенсивність пасажиропотоку (кількість пасажирів, що перевозяться в одиницю часу) у певний час доби обчислюється за такою формулою:

$$S=120 + 1750 \cdot e^{-x} \cdot \sin(x),$$

де $x = 3{,}14\cdot(t-6) / 9$;

t – час доби, протягом якого характеризується пасажиропотік.

Кількість пасажирів, що перевозяться за певний час, обчислюється за такою формулою: $Spas = \int_{t1}^{t2} Sdt$, де **t1** і **t2** – початок і кінець часового інтервалу.

Розробити програму, в якій можна розрахувати кількість пасажирів, що перевозяться протягом певного часового інтервалу доби. Розрахувати також загальну кількість пассжиров, що перевозяться за тиждень в інтервалі часу з 6 до 15 години. Побудувати графік інтенсивності пасажиропотоку в інтервалі цього часу.

Виконання:

1. Створити початкову екранну форму згідно з рис. 8.10. На поверхні форми розміщено імена й назви компонентів, які використовуються за умовчуванням. Надалі імена кожного з компонентів не змінюються, а їх назви відповідають діям, із застосуванням яких вони виконуються.

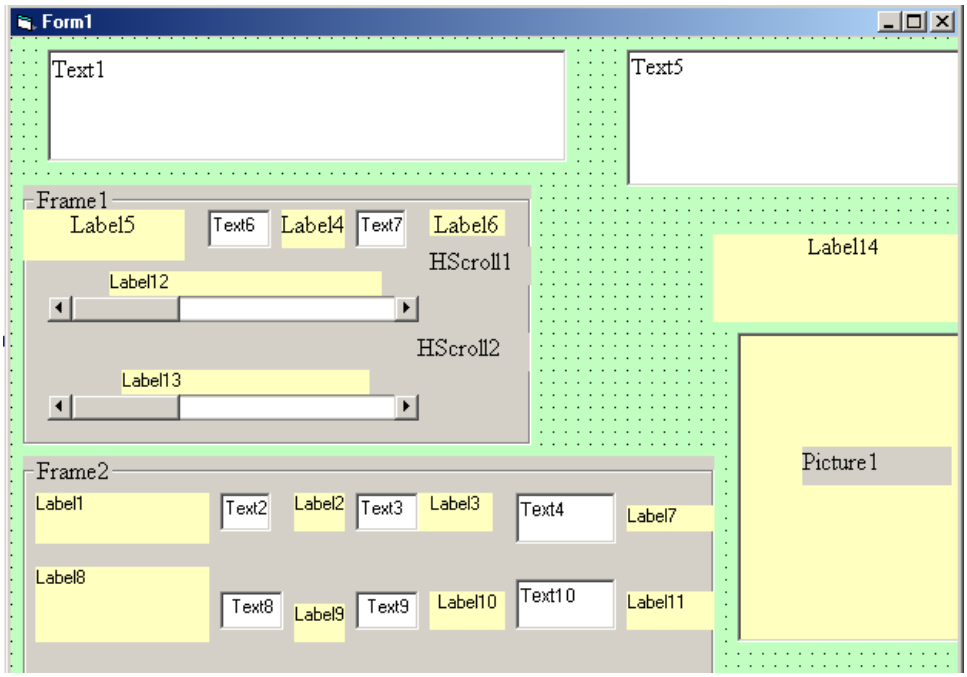


Рис. 8.10. Загальний вигляд початкових даних для створення екранної форми виконання завдання з прикладу 8.3

2. Послідовно виконати операції, спрямовані на створення екранної форми "Розрахунок пасажиропотоку", перелік яких подано в табл. 8.10.

Таблиця 8.10

Порядок створення екранної форми "Розрахунок пасажиропотоку"

Операція	Властивість об'єкта	Значення властивості
1. Дати ім'я електронній формі	**Name**	***Form1***
	Caption	***Розрахунок_ пасажиропотоку***
2. Створити рамку групи інструментів для введення початкових даних (інструмент **Frame)**	**Name**	***Frame1***
	Caption	***Початкові дані***
3. Створити рамку групи інструментів для виведення розрахункових даних (інструмент **Frame)**	**Name**	***Frame2***
	Caption	***Розрахункові дані***
4. Створити текстове вікно для назви лабораторної роботи (інструмент **TextBox**)	**Name**	***Text1***
	Text	***Розрахунок пасажиропотоку на тролейбусному маршруті***
	Multline	***True***
5. Створити текстове вікно для виведення розрахункових даних (початок интервалу) (інструмент **TextBox**)	**Name**	***Text2***
	Text	Порожньо
6. Створити текстове вікно для виведення розрахункових даних (кінець інтервалу) (інструмент **TextBox**)	**Name**	***Text3***
	Text	Порожньо
7. Створити текстове вікно для виведення розрахункових даних (інструмент **TextBox**)	**Name**	***Text4***
	Text	Порожньо
8. Створити текстове вікно для введення назви навчального закладу, інструмент **TextBox**)	**Name**	***Text5***
	Text	Порожньо
	Multline	***True***

Продовження табл.8.10

9. Створити текстове вікно для введення початкових даних (інструмент **TextBox**)	**Name**	***Text6***
	Text	Порожньо
10. Створити текстове вікно для введення початкових даних (інструмент **TextBox**)	**Name**	***Text7***
	Text	Порожньо
11. Створити текстове вікно для виведення розрахункових даних тиждень (інструмент **TextBox**)	**Name**	***Text8***
	Text	***6***
12. Створити текстове вікно для виведення розрахункових даних за тиждень (інструмент **TextBox**)	**Name**	***Text9***
	Text	***15***
13. Створити текстове вікно для виведення розрахункових даних за тиждень (інструмент **TextBox**)	**Name**	***Text10***
		Порожньо
14.Створити лінійку прокручування для введення початку часового интервалу руху (інструмент **HScrollBar)**	**Name**	***HScroll1***
	Large Change	***1***
	Max	***15***
	Min	***6***
15. Створити лінійку прокручування для введення кінцевого часу интервалу руху (інструмент **HScrollBar)**	**Name**	***HScroll2***
	Large Change	***1***
	Max	***15***
	Min	***6***
16.Створити етикетку (інструмент **Label)**	**Name**	***Label1***
	Caption	***Число перевезених пасажирів з***
17. Створити етикетку (інструмент **Label)**	**Name**	***Label2***
	Caption	***до***
18. Створити етикетку (інструмент **Label)**	**Name**	***Label3***
	Caption	***години становить***
19. Створити етикетку (інструмент **Label)**	**Name**	***Label4***
	Caption	***до***

Продовження табл.8.10

20. Створити етикетку (інструмент **Label)**	**Name**	***Label5***
	Caption	***Інтервал часу з***
21. Створити етикетку (інструмент **Label)**	**Name**	***Label6***
	Caption	***години***
22. Створити етикетку (інструмент **Label)**	**Name**	***Label7***
	Caption	***пасажирів***
23. Створити етикетку (інструмент **Label)**	**Name**	***Label8***
	Caption	***Число перевезених пасажирів за тиждень в інтервалі часу з***
24. Створити етикетку (інструмент **Label)**	**Name**	***Label9***
	Caption	***до***
25. Створити етикетку (інструмент **Label)**	**Name**	***Label9***
	Caption	***Інтервал часу з***
26. Створити етикетку (інструмент **Label)**	**Name**	***Label10***
	Caption	***години становить***
27. Створити етикетку (інструмент **Label)**	**Name**	***Label11***
	Caption	***пасажирів***
28. Створити етикетку (інструмент **Label)**	**Name**	***Label12***
	Caption	***Введіть початковий час інтервалу***
29. Створити етикетку (інструмент **Label)**	**Name**	***Label13***
	Caption	***Введіть кінцевий час інтервалу***
30. Створити етикетку (інструмент **Label)**	**Name**	***Label14***
	Caption	***Графік інтенсивності пасажиропотоку з 6 до 15 години, люд.–год***

Закінчення табл.8.10

31. Створити вікно для побудови графіка інтенсивності пасажиропотоку (інструмент **PictureBox)**	**Name**	***Picture1***
	AutoRedraw	***True***
32. Створити командну кнопку для виконання розрахунку інтенсивності вантажопотоку (інструмент **CommandButton)**	**Name**	***Command1***
	Caption	***Розрахунок пасажиропотоку***
33. Створити командну кнопку для очищення полів (інструмент **CommandButton)**	**Name**	***Command2***
	Caption	***Очистити вікна***
34. Створити командну кнопку для виходу з програми (інструмент **CommandButton)**	**Name**	***Command3***
	Caption	***Вихід***

```
Sub SPicture1()
  Cls ' очищення поверхні вікна
  Picture1.Scale (4, 700)-(15, -50)  ' задання розмірів графічного поля і
                                     '  напрямку 'координатних осей
  s2 = 0
  Max = -1000
' Формування циклу для побудови графіка функції інтенсивності
' пасажиропотоку залежно від часу роботи
    For t2 = 6 To 15
        X2 = 3.14152 * (t2 - 6) / 9
        s2 = 120 + 1750 * Exp(-X2) * Sin(X2)
        X21 = 3.14152 * ((t2 + 1) - 6) / 9
        s21 = 120 + 1750 * Exp(-X21) * Sin(X21)
    Next t2
    ' Оцифрування шкали X
    Picture1.PSet (6, 100)
    Picture1.Print "6"
```

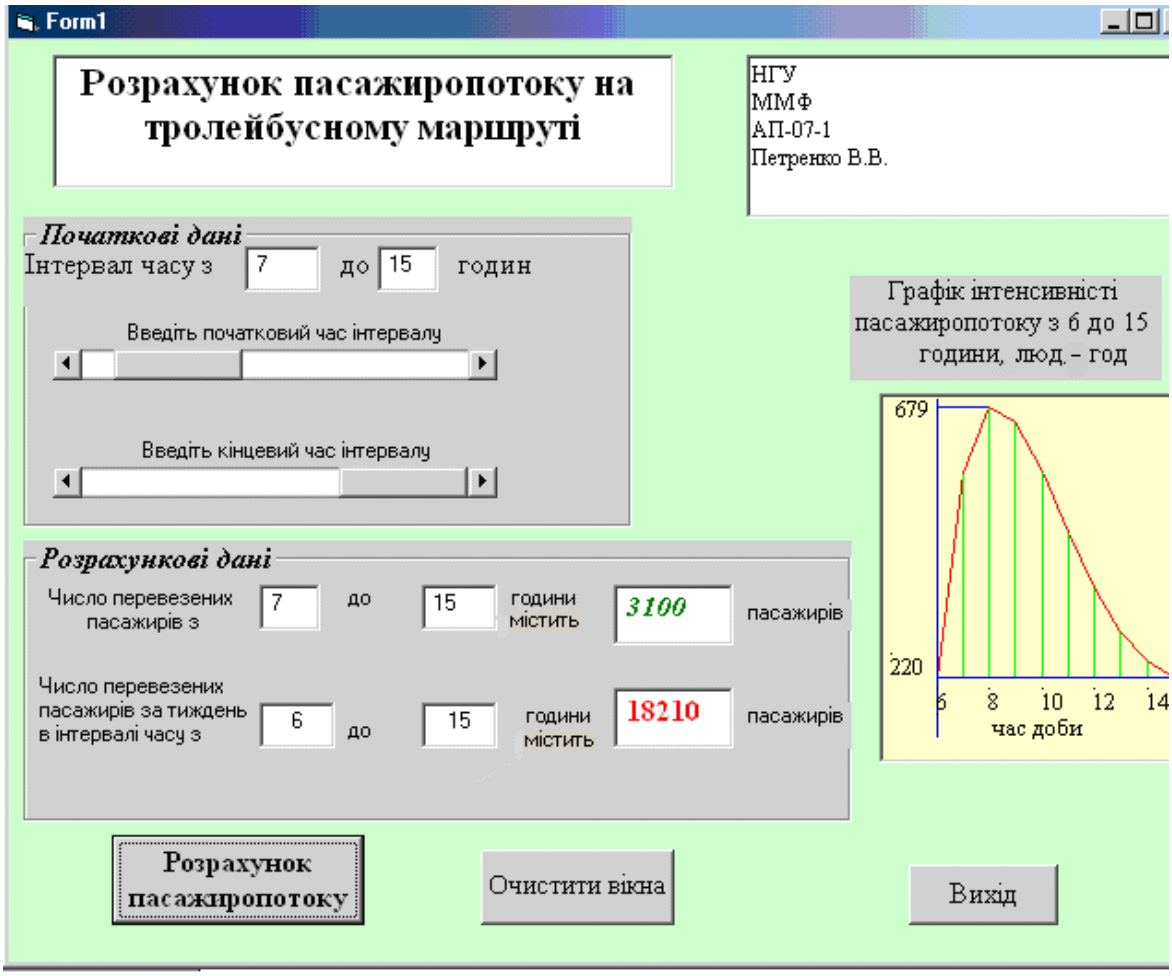


Рис. 8.11. Остаточний варіант екранної форми для розрахунку пасажиропотоку на тролейбусному маршруті

```
    Picture1.PSet (8, 100)
    Picture1.Print "8"
    Picture1.PSet (10, 100)
    Picture1.Print "10"
    Picture1.PSet (12, 100)
    Picture1.Print "12"
    Picture1.PSet (14, 100)
    Picture1.Print "14"
    Picture1.PSet (6, 45)
    Picture1.Print "        час доби"
    Picture1.PSet (4.3, 170)
  ' Позначення мінімальних і максимальних значень інтенсивності
  ' пасажиропотоку
    Picture1.Print "220"
    Picture1.PSet (4.3, (25 + Max))
    Picture1.Print Int(Max)
End Sub
```

```
' Процедура очищення вікон
Private Sub Command2_Click()
   Text6.Text = ""
   Text7.Text = ""
   Text2.Text = ""
   Text3.Text = ""
   Text4.Text = ""
   Text10.Text = ""
   Text5.Text = ""
   Picture1.Cls
End Sub
' Процедура виходу з програми
Private Sub Command3_Click()
    End
End Sub
' Процедури створення лінійок прокручування і роботи з ними для введення
' початкового і кінцевого часу інтервалу
Private Sub HScroll2_Scroll()
    HScroll2_Change
End Sub

Private Sub HScroll2_Change()
     lb2 = HScroll2
    Text7.Text = Str(lb2)
End Sub

Private Sub HScroll1_Scroll()
     HScroll1_Change
End Sub

Private Sub HScroll1_Change()
     lb1 = HScroll1 - 1
     Text6.Text = Str(lb1)
End Sub

' Процедура використання кнопки, Command1  для виконання
' розрахунку обчислення пасажиропотоку
Private Sub Command1_Click()
    enter1 = Chr(13) + Chr(10)
    vus = InputBox("Введіть назву навчального закладу, в якому ви навчаєтесь")
    fak = InputBox("Введіть назву факультету")
    grup = InputBox("Введіть назву групи")
    fam = InputBox("Введіть своє прізвище та ініціали ")
    Text5.Text = vus + enter1 + fak + enter1 + grup + enter1 + fam
```

```
    S = 0: k = 0: s1 = 0
    For i = 1 To 5 ' створення вкладеного циклу для обчислення
        'пасажиропотоку за тиждень у заданому інтервалі
       For t = 7 To 15
          x = 3.14152 * (t - 6) / 9
          S = S + 120 + 1750 * Exp(-x) * Sin(x)
       Next t
     Next i
     Text2.Text = Text6.Text
     Text3.Text = Text7.Text
      l1 = Val(Text6.Text) + 1
      l2 = Val(Text7.Text)
       'створення циклу для
       'обчислення  пасажиропотоку в заданому інтервалі часу доби
      For t1 = l1 To l2 Step 1 ' початок циклу;
          k = k + 1
          X1 = 3.14152 * (t1 - 6) / 9
           s1 = s1 + 120 + 1750 * Exp(-X1) * Sin(X1)
     Next t1   'кінець циклу
    Cls
    Text4.Text = Str(Int(s1))
    Text10 = Str(Int(S))
    SPicture1 ' використання процедури SPicture1 для побудови графіка
              ' інтенсивності пасажиропотоку
End Sub
```

Контрольні питання

1. Яку систему координат прийнято в графіці Visual Basic? Чи можливе її перевизначення?
2. Яким чином встановлюють поточні координати точки, з якої починається графічна побудова об'єкта?
3. Які види графічних примітивів використовуються в Visual Basic.
4. На яких об'єктах Visual Basic можна виконувати графічні побудови?
5. Які способи введення тексту використовуються на графічній поверхні Visual Basic?
6. Назвіть способи форматування тексту в графіці Visual Basic.
7. Які об'єкти можна використовувати для вставлення растрових зображень?
8. У чому полягаєвідмінність між об'єктами Image та PictureBox?
9. Які способи зміни кольору графічних примітивів застосовуються в Visual Basic?
10. Опишить способи зміни типу і товщини ліній у графічних зображеннях Visual Basic.
11. Яким чином можна очистити екран графічної поверхні?

ЛІТЕРАТУРА

1. Бусигін, Прикладна інформатика: Підручник для студентів комп'ютерних спеціальностей / Б.С. Бусигін, Г.М. Коротенко, Л.М. Коротенко. – Національний гірничий університет, 2004. – 559 с.: іл.

2. Войтюшенко, Інформатика і комп'ютерна техніка: Навч. пос. з баз. підготовки для студ. екон. і техн. спеціальностей ден. і заоч. форм навчання / Н.М. Войтюшенко, А.І. Остапець. – К.: Центр навчальної літератури, 2006 – 568 с.

3. Вычислительная техника и программирование: учебник для техн. вузов / А.В. Петров, В.Е. Алексеев, А.С. Ваулин и др.; под ред. А. В. Петрова. – М.: Высшая школа, 1990. – 479 с.: ил.

4. Глинський, Практикум з інформатики: навч. посібник 8-е вид. переробл. і доп. / Я.М. Глинський. – Л.: Деол, СПД Глинський, 2005. – 296 с.: іл..

5. Глинський, Бейсик, Visual Basic і VBA. 4-те вид.перероб. і доп. / Я.М. Глинський, В.Є. Анохін, В.А. Ряжська. – Л.: Деол, СПД Глинський, 2004. – 160 с.

6. Глушаков, Программирование на Visual Basic 6.0: учеб. курс / С.В. Глушаков, С.А. Сурядный; Худож. оформ. А.С. Юхтман. – Х.: Фолио, 2004. – 497 с.

7. Дибкова, Інформатика і комп'ютерна техніка: навч. посіб. / Л.М. Дибкова. – вид. 2-ге, перероб. і доп. – К.: Академвидав, 2005. – 416 с.: іл.

8. Информатика. Базовый курс. / под ред. С.В. Симоновича. – 2-е изд. – С.Пб.: Питер. 2007. – 640 с.: ил.

9. Ковтанюк, Самовчитель роботи на ПК / Ю.С. Ковтанюк, Ю.О. Шпак. –К.: МК-Прес, 2005. – 544 с.: іл.

10. Культин, Visual Basic. Освой самостоятельно / Н.Б. Культин – С.Пб.: БХВ-Петербург, 2005. – 480 с. ил.

11. Сафронов И. К. Visual Basic в задачах и примерах. – С.Пб.: БХВ-Петербург, 2007. - 400 с.: ил.

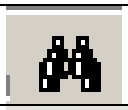
12. Симонович, Компьютер и уход за ним: практ. рук. по эффект. обслуж. комп. / С.В. Симонович, Г.А. Евсеев. – М.: АСТ-ПРЕСС КНИГА; Изд. Развитие, 2004. – 576с .: ил.

13. Слепцова, Программирование на VBA. Самоучитель. / Л.Д. Слепцова. – М.: Издательский дом "Вильямс", 2004. – 384 с. : ил.

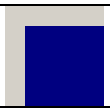
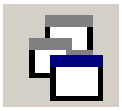
15. Ярмуш, Інформатика і компьютерна техніка: Навч. пос. / О.В. Ярмуш, М.М Редько. – К.: Вища освіта, 2006. – 359 с.: іл.

Додаток 1

Елементи панелі інструментів **Standard** у програмі Microsoft Visual Basic

Кнопка	Назва	Призначення
	***Add Standard EXE Project***	Створення нового проекту
	***Add Form***	Додавання в проект нової форми або файлу
	***Menu Editor***	Служить для створення і редагування меню у відкритій формі
	***Open Project***	Відкриття існуючого проекту
	***Save Project***	Збереження файлів, що входять до складу відкритого проекту
	***Cut***	Вирізування виділеного об'єкта або тексту і занесення його в буфер обміну
	***Copy***	Копіювання тексту або об'єкта в буфер
	***Paste***	Вставлення об'єкту або тексту з буфера обміну в місце розташування курсору
	***Find***	Пошук фрагмента тексту в коді програми
	***Undo***	Відміна останньої виконаної дії в середовищі розробки
	***Redo***	Повторення останньої відміненої дії
	***Start***	Запуск додатка на виконання
	***Break***	Переривання виконання програми (пауза)

Закінчення додатку 1

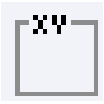
Кнопка	Назва	Призначення
	***End***	Завершення роботи додатка
	***Project Explorer***	Відображення складників відкритого проекту
	***Properties Window***	Вікно редагування властивостей виділеного елементу
	***Form Layout Window***	Вікно розташування форми в додатку, із залученням для наочності маленької моделі екрана
	***Object Browser***	Відображення списку доступних об'єктів у даному проекті, а також швидке переміщення за кодом програми
	***Toolbox***	Відображення панелі інструментів у середовищі розробки

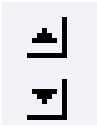
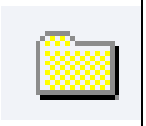
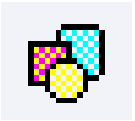
Додаток 2
Елементи панелі інструментів **Form Editor** у програмі Microsoft VisualBasic

Кнопка	Назва	Призначення
	***Bring To Front***	Переміщення виділеного об'єкта на передній план і розташування його перед рештою всіх об'єктів
	***Send To Back***	Переміщення вибраного об'єкта на задній план
	***Align Lefts***	Вирівнювання при потребі кількох виділених об'єктів форми вирівняти певним способом по відношенню до якого-небудь з них, причому виділення еталонного об'єкта відбувається в останню чергу
	***Center Horizontally***	Розміщення виділеного об'єкта в центрі форми порівняно з вертикальною та горизонтальною осями
	***Make With Same Size***	Зміна розмірів усіх вибраних об'єктів на формі так, щоб їх висота і/або ширина дорівнювали, відповідно, висоті і/або ширині того об'єкта, що був виділений останнім
	***Lock Controls Toggle***	Заборона на переміщення об'єкту формою або на зміну його розмірів

Додаток 3

Опис вбудованих елементів управління у програмі Microsoft Visual Basic

Елемент	Назва	Призначення
	***Pointer*** *(покажчик)*	Вибір елементів керування
	***PictureBox*** *(картинка)*	Виведення графічних елементів у формі (може використовуватися як *контейнер)*
	***Label*** *(мітка)*	Відображення написів у формі
	***TextBox*** *(текст)*	Уведення тексту
	***Frame*** *(рамка)*	Об'єднання в групу різних елементів керування (такі об'єкти називаються *контейнерами)*
	***Command Button*** *(кнопка, що управляє)*	Виконання додатком певних дій, викликаних натисненням на керуючу кнопку
	***Checkbox*** *(прапорець)*	Установлення/відключення налаштувань
	***OptionButton*** *(перемикач)*	Вибір користувачем одного з кількох можливих пунктів (їх повинно бути не менше двох)
	***ComboBox*** *(комбінований список)*	Вибір елемента із розкривного списку
	***ListBox*** *(список елементів)*	Вибір користувачем якого-небудь елемента з тих, що є в списку
	***HscrollBar*** *(горизонтальна смуга прокручування)*	Перегортання в горизонтальному напрямку списку, якій міститься в іншому елементі керування списком

Елемент	Назва	Призначення
	***VscrollBar*** *(вертикальна смуга прокрутки)*	Перегортання у вертикальному напрямку списку, якій міститься в іншому елементі керування списку
	***Timer*** *(таймер)*	Виконання додатком дій у реальному часі
	***DriveListBox*** *(список дисків)*	Вибір користувачем якого-небудь диска
	***DirListBox*** *(список каталогів)*	Вибір користувачем каталога на диску
	***FileListBox*** *(список файлів)*	Вибір користувачем файлу з каталога
	***Shape*** *(фігура)*	Відображення геометричних фігур у формі
	***Line*** *(лінія)*	Побудова графічних ліній
	***Image*** *(зображення)*	Виведення графічних елементів у формі (не може використовуватися як *контейнер)*
	***Data*** *(дані)*	З'єднання з існуючою базою даних
	***OLE***	Уведення в додаток функцій інших програмних засобів

Навчальне видання

Дудко Михайло Олександрович
Мацюк Ірина Миколаївна
Вернер Ілля Володимирович

КОМП'ЮТЕРНА ТЕХНІКА ТА ПРОГРАМУВАННЯ
Навчальний посібник

Редактор О.Н. Ільченко

Підписано до друку 29.04.2010. Формат 30х42/2.
Папір офсетний. Ризографія. Ум., друк. арк. 9,6
Облік.-вид. арк. 12,6 Тираж 300 прим. Зам.

Підготовлено до друку та видруковано
у Національному гірничому університеті.
Свідоцтво про внесення до Державного реєстру ДК № 1842.
49005, м. Дніпропетровськ, просп. К. Маркса, 19.