# Конспект лекцій

**З М І С Т**

# В С Т У П

Пропонований конспект лекцій відповідає робочій програмі дискипліни «Інформатика та комп'ютерна техніка» для студентів 1-го курсу товарознавчого факультету. Орієнтовано конспект лекцій на одержання знань за темою «Основи програмування мовою Visual Basic» для тих, хто вже має елементарні знання про будову ПК, про операційні системи та принципи роботи на ПЕОМ.

Даний конспект лекцій знайомить студентів з основами алгоритмізації розв'язків фахових задач та з їх програмуванням в середовищі програмування Visual Basic. За допомогою матеріалу, який викладено, студенти мають засвоїти не тільки елементарні основи програмування з застосуванням простих даних, а також складні та цікаві прийоми та методи програмування з використанням процедур, функцій, масивів та файлів даних.

Структурно конспект включає 7 тем (лекцій):

1. Алгоритми обчислюваних процесів
2. Програмування на алгоритмічних мовах
3. Основи системи програмування Visual Basic
4. Оператори управління
5. Масиви
6. Модульне прогамування
7. Робота з файлами

Весь матеріал конспекту подано в єдиному форматі. Кожна лекція закінчується контрольними запитаннями для закріплення матеріалу.

# *Лекція* 1

# АЛГОРИТМИ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

***План лекції:***

*1.1. Етапи розв'язування задачі на ЕОМ*

*1.2. Поняття алгоритму*

*1.3. Властивості алгоритмів*

*1.4. Види алгоритмів*

## 1.1. ЕТАПИ РОЗВ'ЯЗУВАННЯ ЗАДАЧІ НА ЕОМ

Від часу створення першої обчислювальної машини минуло більше п'яти десятиліть. За цей час кілька разів змінювалася елементна база ЕОМ, зменшилися розміри і споживані потужності, збільшилась швидкість обчислень, стало набагато зручніше з ними працювати. Впровадження й широке використання засобів обчислювальної техніки є одним з головних факторів прискорення науково-технічного прогресу в будь-якій країні світу. Стрімко зростає роль ЕОМ у всіх областях людської діяльності. Без використання швидкодіючих ЕОМ немислиме рішення завдань інтенсифікації економічного розвитку провідних галузей народного господарства.

Темпи науково-технічного прогресу, посилення ролі в значній мірі визначаються якістю й номенклатурою засобів обчислювальної техніки і їхнім програмним забезпеченням. Саме розвиток цих засобів забезпечує успіхи в автоматизації виробничих процесів, у розробці нових технологій, у підвищенні ефективності праці й керування. Широке й різноманітне застосування ЕОМ ставить усе більше високі вимоги до їхнього програмного забезпечення. Розробка програм і програмних комплексів здобуває характер індустріального виробництва. Значення програмного забезпечення важко переоцінити, тому що саме програми визначають і створюють «інтелект» комп'ютера. У той же час процес створення програм ставиться до однієї з найбільш складних сфер творчої діяльності людини, що вимагає більших зусиль і спеціальної технології розробки.

Не дивлячись на те, що номенклатура програмного забезпечення складає десятки тисяч програм, які торкаються всіх сторін людської діяльності, кожен фахівець може знайти в своїй області задачу, розв'язок якої хотів би автоматизувати.

Будь-яка задача, перш ніж вона може бути розв'язана на ЕОМ, проходить підготовчий шлях, який складається з декількох етапів.

- Спочатку потрібно сформулювати задачу, з'ясувати що вимагається знайти та що для цього відомо.

- Після цього необхідно подати математичне описання задачі. Для цього слід призначити імена відомим і невідомим змінним та постійним величинам, що характеризують явище, знайти формули чи рівняння, що їх зв’язують, або задати ці зв’язки у вигляді таблиць чи графіків. Це називають *побудовою математичної моделі явища* та *формалізацією задачі.*
- Далі розробляється або обирається з множини відомих метод розв'язування даної задачі та алгоритм його реалізації.
- На наступному етапі розробляється та налагоджується програма на алгоритмічній мові, або вибирається з множини вже існуючих програм.
- На етапі налагодження програми переконуються, що вона не містить синтаксичних і логічних помилок, застосувавши її до тестових завдань з заздалегідь відомими результатами.
- Нарешті програма запускається в дію, вводяться початкові дані і знаходяться результати, які аналізуються користувачем.

Важливим етапом є розробка алгоритму розв’язування задачі.

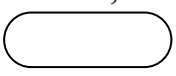
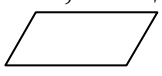
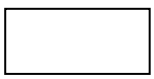
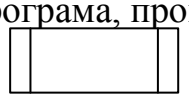
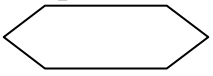
## 1.2. ПОНЯТТЯ АЛГОРИТМУ

*Алгоритм* – це послідовність дій над початковими даними та проміжними результатами необхідних для одержання кінцевого результату.

Алгоритм можна описати словами, у вигляді графічної схеми та програми на одній з мов програмування.

Графічне зображення алгоритму або блок-схема складається з блоків (графічних символів), що зв'язуються між собою направленими лініями. Читається схема зверху вниз, та зліва направо, тому стрілки, що вказують напрямок ліній можуть бути відсутніми.

Найбільш часто зустрічаються наступні блоки

| | |
|---|---|
| Початок, кінець  | Введення, виведення  |
| Обчислення, процес  | Логічний блок, перевірка умови  Так Ні |
| Підпрограма, процедура  | Блок модифікації циклу  |
| Сполучник  | Міжсторінковий сполучник  |

## 1.3. ВЛАСТИВОСТІ АЛГОРИТМІВ

Алгоритм повинен мати наступні властивості:

*Дискретність* – процес рішення подається як послідовність кроків (етапів) і відбувається в часові дискретно.

*Детермінованість*(визначеність) – кожен крок повинен бути чітко визначеним і не повинен допускати довільного тлумачення.

*Результативність* – алгоритм повинен приводити до рішення задачі за скінчене число кроків.

*Масовість* – алгоритм рішення задачі розробляється в загальному виді, таким чином, щоб він міг бути застосованим для цілого класу задач, що різняться лише початковими даними.

## 1.4. ВИДИ АЛГОРИТМІВ

Розрізняють лінійні, розгалужені та циклічні алгоритми. Алгоритм вирішення будь-якої задачі може бути поданий як комбінація цих трьох вище названих.

**Лінійний алгоритм** характеризується одно направленим послідовним переходом від блоку до блоку по мірі їх виконання.

**Приклад 1.** *Обчислити очікуване накопичення на внесок 2000 грн., який було зроблено клієнтом банку під 12 відсотків терміном на 5 років.*

Формалізуємо постановку задачі. Позначимо очікуване накопичення змінною *S*, величину внеску – *B*, річний відсоток на внески – *p*, термін внеску – *n*.

Тоді результат можна обчислити за формулою $S=B(1+p/100)n$.

Блок-схему алгоритму обчислень відображено на рис. 1.1.

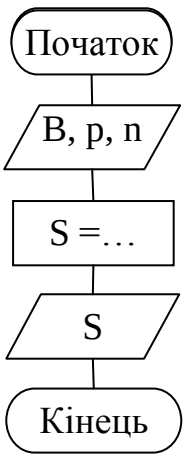



**Рисунок 1.1.** Блок-схема лінійного алгоритму

**Розгалужений алгоритм** має місце в тому разі, коли в залежності від виконання чи невиконання умови виконується одна чи друга дія.

**Приклад 2**. *Визначити суму доплат за роботу в нічний час за формулою*

$$S = \begin{cases} 0.5 \cdot T \cdot t_n, & \text{якщо} \quad t_n \le 2 \\ T + T \cdot (t_n - 2), & \text{якщо} \quad t_n > 2 \end{cases}$$

*де Т-тарифна ставка, а $t_n$ – час відпрацьований вночі.*

Блок-схема алгоритму може мати наступний вид

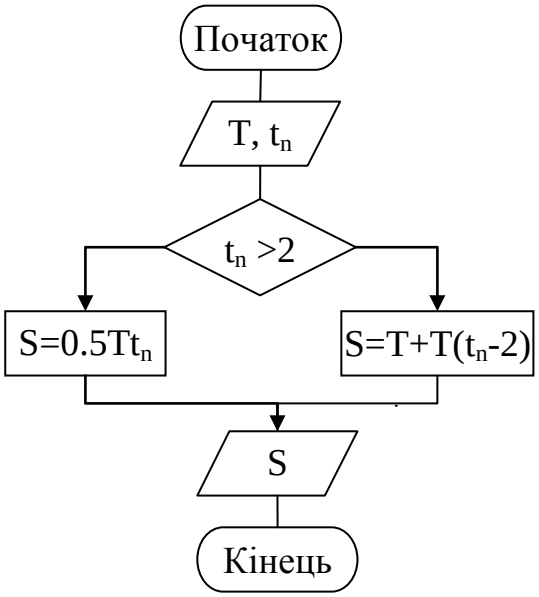



**Рисунок 1.2.** Приклад алгоритму розгалужених обчислень

**Множинний вибір** є узагальненням розгалуження, коли в залежності від значення змінної (I) виконується одна з багатьох дій.

**Приклад 3.** *Обчислити денний заробіток робітника з погодинною оплатою враховуючи його розряд, якщо самі тарифи можуть змінюватися але коефіцієнти розрядної сітки остаються незмінними і визначаються таблицею*

| Розряд | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Коефіцієнт | 1 | 1.1 | 1.35 | 1.5 | 1.7 | 2 | 2.2 |

Позначимо денний заробіток літерою *Z*, розряд – *r*, кількість відпрацьованих робітником годин – *n*, тариф – *T*, коефіцієнт тарифної сітки – *k*. Тоді *Z=k*n*T*, де значення *k* залежить від значення *r*.

Блок-схема алгоритму рішення задачі з використанням оператора множинного вибору відображена на рис. 1.3

**Циклічний алгоритм** передбачає багаторазове повторення якихось дій (обчислень, введення даних, виведення результатів, то що) при різних початкових умовах.

Цикли для яких число повторень наперед відоме або легко визначається називають *арифметичними***.**

Цикли в яких число повторень невідоме, а повторення припиняються як тільки внаслідок них досягається виконання певної умови (умови виходу з циклу) називають *ітераційними*.

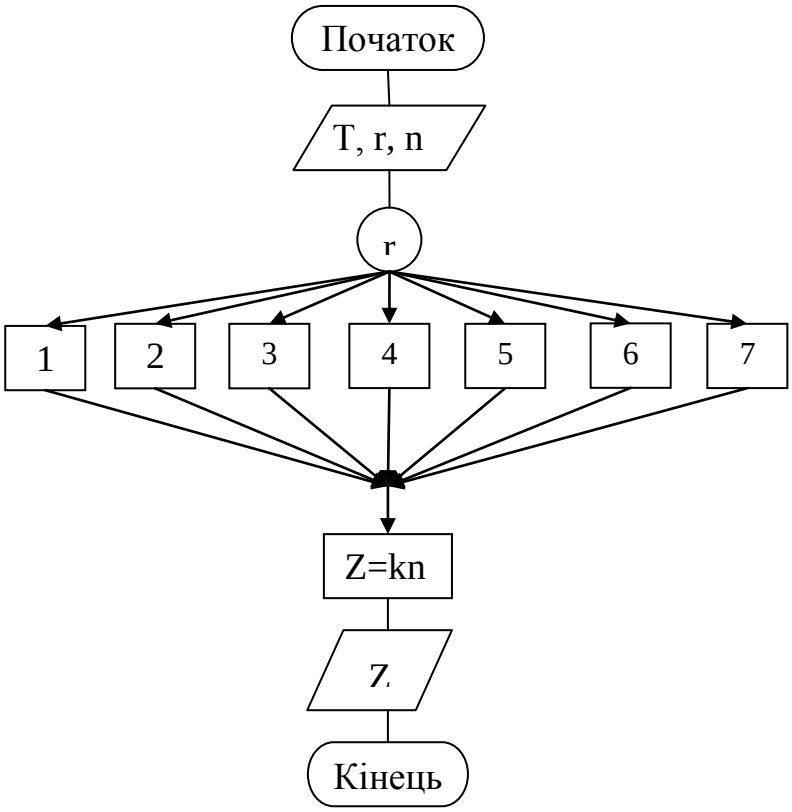



**Рисунок 1.3.** Блок-схема рішення задачі 3

Оператори які повторюються в циклі створюють тіло циклу. Перевірка умови може виконуватись до початку виконання тіла циклу (цикл типу **Поки**) або після виконання операторів тіла (цикл типу **До**).

Розглянемо приклад застосування арифметичного циклу.

**Приклад 4.** *Використовуючи результат прикладу 3, підрахувати денні заробітки робітників цеху і сумарні витрати на оплату їх праці.*

Особу робітника будемо визначати за його кодом-номером *i* у списку робітників. Для забезпечення масовості алгоритму позначимо число робітників літерою *n*. На початку рішення вкажемо це число, потім задаємо код робітника (почнемо з *i*=1). Витрати на оплату праці позначимо *S*. До початку обчислень вважатимемо *S*=0.

Нам потрібно *n* раз повторити одну і ту ж сукупність дій – обчислення заробітку робітника за його розрядом і тарифом, як це зроблено в прикладі 3. Щоб позбутись повторень, відповідну групу дій можна відобразити на блок-схемі окремим блоком і звертатись до нього, коли виникає потреба. Відособлена група операторів, яку можна повторювати багаторазово, звертаючись до неї з різних місць

програми називається підпрограмою або процедурою. Щоб підпрограма при звертанні до неї виконувалась кожен раз з новими даними, її треба складати в загальному вигляді, а вихідні дані для роботи передавати в змінні підпрограми перед звертанням до неї.

Тож звертаємось до алгоритму створеного в прикладі 3 як до процедури *Z*. Оскільки робітники можуть мати різні професії, то введення значень тарифу, крім розряду і тарифного коефіцієнту, на кожному кроці оправдане. Отримане значення заробітку *Z* додаємо до *S*, накопичуючи суму, збільшуємо параметр циклу *i* (код робітника) на одиницю, перевіряємо умову *i>n* (умова того, що враховані заробітки всіх робітників цеху).

В разі її виконання припиняємо обчислення.

Блок-схема розглянутого алгоритму зображена на рис.1.4.

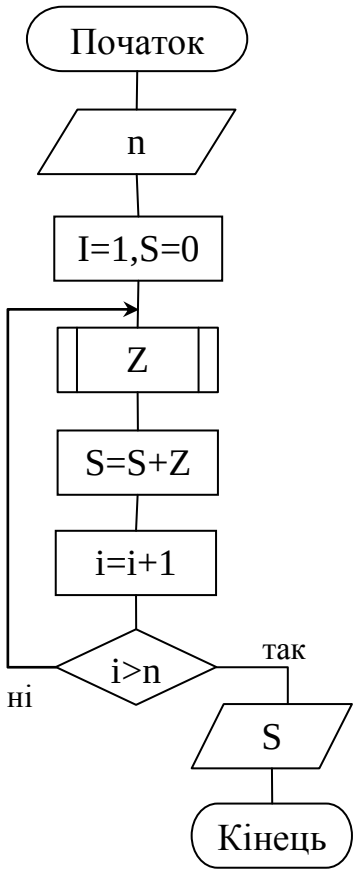
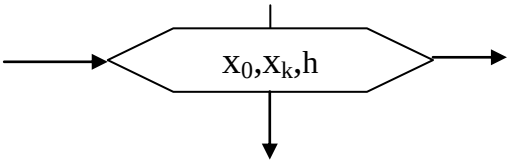



**Рисунок 1.4.** Блок-схема до Прикладу 4

На рис. 1.5 відображена блок-схема того ж алгоритму але з використанням модифікованого блоку циклу.

Модифікований блок циклу має вигляд




Текст, що відображено в блоці сповіщає про те, що параметр циклу змінюється від початкового значення $x_0$ до кінцевого $x_k$ з кроком $h$. Якщо крок дорівнює 1, то його можна не вказувати.

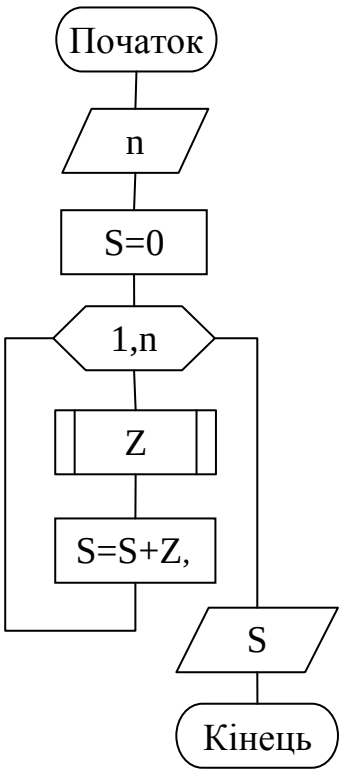



**Рисунок 1.5.** Блок-схема до Прикладу 4

**Приклад .** *Створити двомірний масив розміром m*n, знайти суму його додатних елементів, та знайти найбільший елемент масиву.*

Для введення елементів матриці використовуємо подвійний цикл. Параметр зовнішнього циклу – номер рядка *i*, що змінюється від 1 до *m*, параметр вкладеного циклу – номер стовпця *j*, що змінюється від 1 до *n*.

Для обчислення суми *S* додатних елементів матриці спочатку надаємо змінній *S* значення 0 (*S*=0). Потім в подвійному циклі елементи матриці перевіряємо на виконання умови $a_{ij}>0$ і в разі *так*, накопичуємо суму $S=S+a_{ij.}$

Для знаходження найбільшого елемента спочатку зробимо припущення що $Max=a_{11}$, а потім в подвійному циклі порівнюємо *Max* з наступними елементами і якщо виявиться більший, то його оголосимо максимальним, після чого продовжимо порівнювання. При виході з циклу *Max* буде найбільшим елементом масиву. Для більшої компактності алгоритму можемо об’єднати накопичення суми і пошук максимуму з процесом введення елементів масиву. Алгоритм розв’язування

задачі відображено на рис. 1.11.

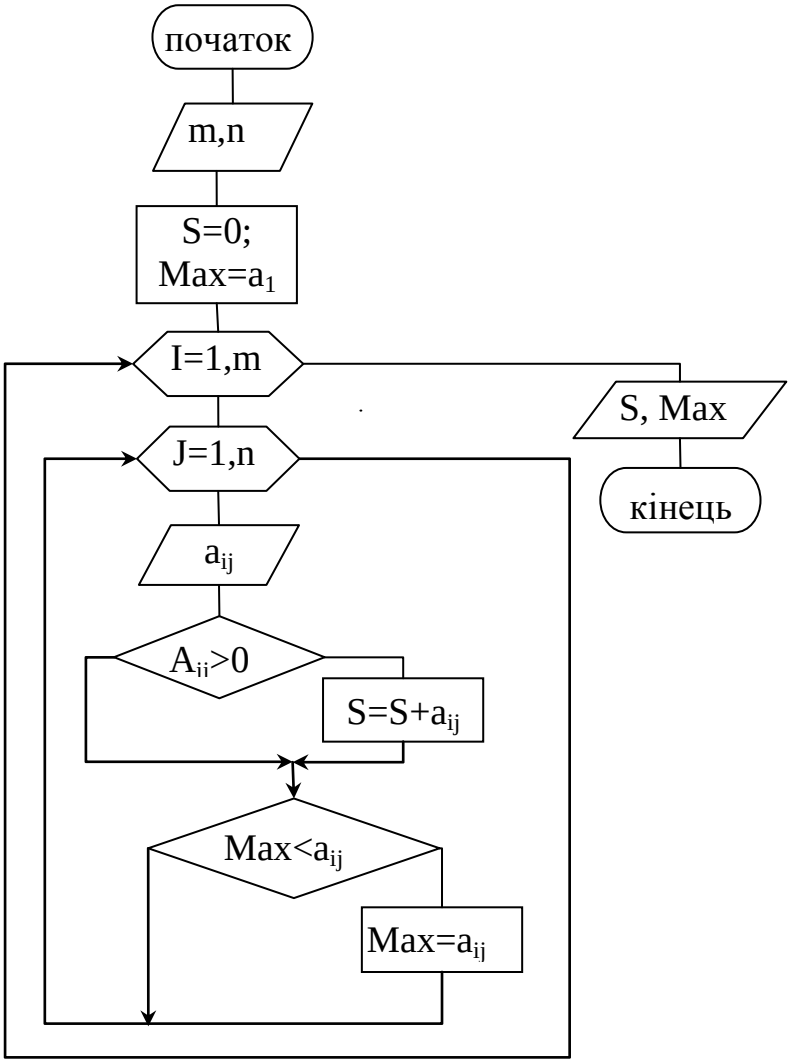



**Рисунок 1.11.** Блок-схема алгоритму обчислення суми елементів матриці та знаходження її найбільшого елемента.

На сьогодні існують технології розробки програм без використання блок-схем. Однак незалежно від цього на початковому етапі вивчення програмування доцільно при розробці алгоритму їх використовувати.

Найбільш поширеним описом алгоритмів рішення задач є програми написані на алгоритмічних мовах. Алгоритмічні мови високого рівня наближені до натуральної англійської мови, але мають обмежену кількість слів з яких будуються речення-рядки програм за строго визначеними правилами. Програма, складена на алгоритмічній мові, не може виконуватись ЕОМ безпосередньо, оскільки ЕОМ може виконувати тільки послідовності елементарних операцій, а команда в програмі на алгоритмічній мові може потребувати виконання десятків і сотень елементарних операцій, що зрозуміло людині, та недоступно ЕОМ. Переведення про-

грами з алгоритмічної мови на машинну здійснюється самою ЕОМ за допомогою спеціальної програми, що називають транслятором. В програмі – трансляторі закладені всі правила алгоритмічної мови і способи перетворення різних конструкцій мови в послідовності елементарних команд зрозумілих машині. Тобто транслятор перетворює програму написану на алгоритмічній мові в програму на машинній мові. Найбільш поширеними мовами для ПК в наш час є Паскаль, Бейсик, C++, Java.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. З яких етапів складається рішення задачі на ЕОМ?
2. Що таке алгоритм та яким чином можна його описати?
3. Що таке схема алгоритмів?
4. З яких блоків може складатися блок-схема алгоритму?
5. Яким чином читається алгоритм? Чи обов'язкова наявність стрілочок у графічній схемі алгоритму?
6. Які властивості повинен мати алгоритм?
7. Які існують правила розробки алгоритмів?
8. Які різновидності структур алгоритмів ви знаєте. Наведіть приклади.
9. Які цикли називаються арифметичними? Наведіть приклади.
10. Які цикли називаються ітераційними? Наведіть приклади.
11. Чим відрізняються лінійний, циклічний та розгалужений алгоритми?
12. Про який алгоритм можна сказати, що він має складену структуру?
13. Що таке масив? Які види масивів ви знаєте?
14. Надайте приклад одно- та двовимірного масивів.
15. Який опис алгоритмів рішення задач ви знаєте? Що таке алгоритмічна мова?

# *Лекція* 2

# ПРОГРАМУВАННЯ НА АЛГОРИТМІЧНИХ МОВАХ

***План лекції:***

*2.1. Поняття про мови програмування*
*2.2. Система програмування Visual Basic*
*2.3. Поняття проекту Visual Basic*
*2.4. Інтегроване середовище розробки (IDE)*
*2.5. Вікно форми та його властивості*
*2.6. Основні елементи керування*

## 2.1. ПОНЯТТЯ ПРО МОВИ ПРОГРАМУВАННЯ

В ЕОМ вся інформація відображається у двійковій формі, а для виконання обчислень використовується двійкова система числення.

Це означає, що в комп`ютері для зберігання й обробки інформації використовується алфавіт, який складається тільки з двох символів: 0 та 1. Використання двійкової системи числення значно спрощує технічні засоби, підвищує надійність їх роботи, хоч і збільшує кількість розрядів, необхідних для запису числа.

У двійковому коді записуються і числа, і коди операцій, і цілі тексти. ЕОМ реагує тільки на послідовність імпульсів (змін рівня напруги), що розглядаються як 0 та 1. Ці послідовності називають машинною мовою. Саме такі послідовності записуються у пам`ять машини.

Процесор здійснює вибірку команд з пам`яті, їх декодування та виконання. Людині було б важко писати та читати програми в двійкових кодах, тому було вигадано більш зручні для людини системи кодування – алгоритмічні мови програмування, які потім самою машиною транслюються у машинні коди. Програма записується людиною на зручній для неї вхідній мові , а транслюється в об`єктний код зрозумілий для машини. Вхідна мова складається з певного набору слів, що вказують на дії. Ці слова називають операторами мови.

Найпростішою з таких мов є *мнемокод*. Його оператори відображують суть операцій, вони являють собою скорочені до 3-4-х символів назви операцій на звичайній мові. Скорочення здійснюється так, щоб залишався зрозумілим сенс слова, це полегшує запам`ятовування та використання мнемокоду.

Для кожного типу машин використовуються свої мнемокоди, тому їх називають *машинно-орієнтованою мовою*. Символічну мову програмування, що складається з мнемонічних скорочень назв машинних команд на англійській мові на-

зивають *мовою Асемблера*, або просто *Асемблером*. Програма, яка транслює написаний на Асемблері текст у машинний код також називається *Асемблером*. Це також *машинно-орієнтована мова*. Більш досконалими машинно-незалежними є мови високого рівня. Мова програмування високого рівня – це штучно створена формальна мова, яка може бути переведена у машинний код і одночасно зберігає схожість з натуральною мовою. Роботи над створенням мов високого рівня почались в США ще в 50-х роках.

У 1954 роках у США було створено *Фортран* (від «Формул транслятор»). Ця мова досить складна, тому на її базі було створено більш просту мову програмування BASIC, яка стала найпопулярнішою мовою програмування. У Європі в кінці 50-х років було створено мову *Алгол*.

Мова *Паскаль* є прямим нащадком Алгола. Вона була створена швейцарським математиком Ніколсом Віртом у 1969 р. Паскаль є дуже компактною мовою, її опис займає усього біля 30 сторінок. Транслятор з Паскаля є простим і займає небагато місця в ОЗП, що зручно для ПК та мікро ЕОМ. Тому Паскаль став таким популярним серед користувачів ПК, як і Basic. На ньому навчають програмуванню в багатьох ВНЗах.

Ми з вами будемо вивчати більш популярну версію мови Basic – *Visual Basic*.

*Basic* був розроблений у 1965 році у Дортмундському коледжі в США викладачем Джоном Кемені, як дуже спрощений варіант мови *Фортран*.

Назва BASIC має потрійну змістовність:

- по перше це абревіатура “Beginners All-Purpose Symbolic Instruction Code”, тобто “Багатоцільова мова символічних інструкцій для початківців”;
- по-друге так називалась колись мова, яка була розроблена для туземного населення колоній Англії, що нараховувала біля 300 слів та дозволяла спілкуватись на будь – яку тему;
- нарешті Basic означає базовий.

Головне достоїнство мови – це її простота, можливість діалогу між людиною і комп`ютером, що дуже допомагає під час роботи з ПК і подобається користувачам. На наш час вже створено до 40 версій мови BASIC. Найбільш потужними є GW BASIC, QBASIC, QuickBasic та Turbo BASIC.

Перші дві є інтерпретаторами, дві другі компіляторами.

Інтерпретатор зчитує рядок програми, транслює його у машинні коди і виконує оператори, що записані в цьому рядку, потім переходить до наступного рядка. Компілятор відразу зчитує всю програму, транслює її у машинний код (компілює) і виконує. Скомпільована програма може бути записана як окремий, гото-

вий для використання бінарний файл. Така програма виконується набагато швидше ніж під інтерпретатором.

У кожній з мов існують як засоби описування дій – *оператори*, так і засоби для спілкування з машиною – *команди*. Основним режимом в усіх версіях є програмний режим, коли загодя написана програма цілком вводиться в ЕОМ, а потім виконується. В іншому режимі, що називають режимом безпосереднього виконання, оператори мови виконуються відразу після вводу в ЕОМ. У більшості версій програми складаються з нумерованих рядків. Звичайно рядки нумеруються, починаючи від 10 з кроком 10. В QBASIC та TB BASIC рядки можна не нумерувати, ставлячи мітки перед тими рядками, на які передбачається передача управління операторами умовного чи безумовного переходів. У тих версіях, де нумерація рядків обов`язкова, рядки можна вводити в будь-якому порядку. Вони однаково будуть виконуватись в напрямку зростання номерів.

В Q- та TB BASIC оператори і команди виконуються в тій послідовності в якій вводяться, якщо порядок не порушується операторами переходів.

Visual Basic, як це витікає з назви мови, реалізує найсучасніший підхід візуального програмування, що дозволяє значно прискорити час розробки професійних програм при мінімумі самого програмування.

Суть візуального програмування в тім, щоб дати вам можливість зосередитися на програмуванні вашого професійного завдання, не витрачаючи маси часу на подання даних на екрані або їхнє збереження на диску.

Саме по собі візуальне програмування – це величезний крок уперед у порівнянні з традиційним. Якщо традиційне програмування являє собою процес кодування, що складається в написанні послідовності команд алгоритмічної мови, то при візуальному програмуванні кодування відсувається на другий план. Спочатку за допомогою миші встановлюються кнопки, прапорці та інші елементи користувальницького інтерфейсу, формуючи зовнішній вигляд програми. Потім задаються їхні властивості й лише в останню чергу, якщо залишилася така необхідність, проводиться написання підпрограм реакції елементів керування на зовнішні події, наприклад щигликів миші або натискань клавіш. Таким чином, основою будь-якого додатку, розробленого в середовищі візуального програмування, є вікно з елементами інтерфейсу, з якими зв'язані окремі підпрограми.

## 2.2. СИСТЕМА ПРОГРАМУВАННЯ VISUAL BASIC

Visual Basic відноситься до групи програмних засобів під загальною назвою системи програмування. Система програмування забезпечує користувача середо-

вищем для розробки програм, в Visual Basic це називається проектуванням додатків. Система програмування Visual Basic вміщує текстовий редактор для друку текстів програм та конструктор форм. Програміст пише вихідні тексти програм формалізованою мовою, що являє собою послідовність команд чи операторів. За допомогою конструктора форм виконується розробка інтерфейсу програми. Visual Basic являється одночасно компілятором і інтерпретатором. Щоб програма виконувалася, вихідні тексти переводяться на машинну мову, це робить компілятор, що також входить у систему програмування. Не виходячи з середовища Visual Basic, ви можете багаторазово запускати свою програму на виконання, перевіряючи і налагоджуючи її роботу, і повертатися назад (інтерпретатор). Таким чином, програма може знаходитися або в режимі проектування, або в режимі виконання.

Система програмування Visual Basic має ряд нових можливостей, таких як:

- використання нової оперативної довідкової системи Microsoft Developer Network (MSDN);
- швидкий запуск проекту одного або більше наперед визначеного формою;
- використання наборів Visual Basic для додатків;
- використання засобів Automation (Автоматизація) Microsoft Word, Microsoft Excel, Microsoft Outlook та Power Point;
- використання засобів управління Multimedia MCI для роботи з мультімедія (використання музикальних ефектів та відео кліпів);
- використання нових функцій Windows API (широкий набір функцій, що дозволяє виконувати повсякденні задачі в операційній системі Microsoft Windows);
- використання Microsoft Internet Explorer для відображення документів HTML та ін.

## 2.3. ПОНЯТТЯ ПРОЕКТУ VB

В Visual Basic всі розроблені додатки називаються проектами. Всі проекти Visual Basic будуються по модульному принципу, тому і об'єктний код складається не з одного великого файлу, а з декількох частин. Проект в Visual Basic складається з багатьох компонент: форм, модулів, класів і ресурсів. Всі ці компоненти об’єднуються в єдиному файлі проекту (VBP). У вікні проекту відображаються всі елементи додатку: форми, модулі тощо, згруповані по категоріям, за виключенням деяких файлів, наприклад, малюнків, довідкових файлів, текстових і деяких інших. Проект утримує декілька груп компонентів:

- файл кожної форми (*. FRM);

- файл кожної форми з елементами управління, який утримує бінарну інформацію (*.FRX);
- файл кожного модуля (*.BAS);
- файл кожного модуля класів (*.CLS);
- файл додаткових елементів управління (*.OCX);
- максимум один файл ресурсів (*.RES);
- файл проекту, який утримує посилання на свої компоненти (*.VBP);
- група проекту(*.VBG);
- ряд додаткових файлів, які залежать від вигляду проекту (*.CTL, *.BMP, *.MDB тощо).

Необхідно пам'ятати деякі особливості збереження проекту та його компонентів Visual Basic. Всі ці елементи зберігаються як окремі і незалежні файли. Тому їх можна в будь-який час завантажувати і зберігати. При натисканні кнопки збереження на панелі інструментів зберігається не весь проект, а тільки активний компонент (модуль або форма). Для того, щоб зберегти відповідний елемент (форму, модуль тощо) також можна виділити його в списку вікна проекту та обрати команду **Сохранить** (**File\Save**) або **Сохранить как** (**File\Save As**). Для збереження всього проекту (включаючи всі компоненти) вибираємо команду **Сохранить Програму** (**File\Save Projec**t) або **Сохранить Програму как** (**File\Save Project As**) (для збереження проекту з іншим ім'ям).

Пам’ятайте, що у файлі компонента не зберігається інформація про те, до якого проекту він належить. Список компонентів проекту та зв'язок між його компонентами зберігаються тільки у файлі проекту (VBP). Декілька проектів можуть бути об'єднані в один файл групи (VBG).

Таким чином, в будь-яких проектах можна використовувати деяку складову частину інших проектів, наприклад форму, модуль тощо. Для цього необхідно скопіювати в каталог проекту необхідний файл і додати їх до проекту командою **Project\Add**...

Щоб додати до проекту новий елемент, необхідно в меню **Разработать** обрати необхідний елемент або за допомогою контекстного меню **Добавить** (у списку, що спливе обрати необхідний елемент). Для видалення елемента його необхідно виділити у вікні проекту і в меню **Разработать** або в контекстному меню обрати команду **Удалить.**

Для того, щоб програма Visual Basic могла виконуватись не тільки в середовищі Visual Basic, необхідно її скомпілювати. Для компіляції необхідно в меню **Файл (File)** вибрати команду **Создать проект: <ім’я проекту> .exe.**

## 2.4. ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРОБКИ (IDE)

Після запуску Visual Basic на екрані з'являється діалогове вікно (рис.2.1), у якому ви можете вибрати тип створюваного додатка. З цього ж вікна можна завантажити вже існуючий проект. За деякими піктограмами діалогового вікна ховаються майстри (Wizards), що супроводжують розроблювача при створенні додатків і що беруть на себе частину його роботи, наприклад підключення бази даних чи створення форми. Один з основних майстрів – майстер додатка Visual Basic, за допомогою якого можна створити основний «каркас» для звичайних Windows-додатків. У процесі роботи майстра створюється майже готовий додаток з різними формами, з відповідним робочим середовищем, меню, панеллю інструментів та ін. Цей додаток можна потім удосконалювати і набудовувати. Якщо ви не маєте достатнього досвіду розробки додатків, то створені вами додатки будуть виглядати як звичайні Windows-додатки. Такий додаток створюється за допомогою елемента Standard EXE.

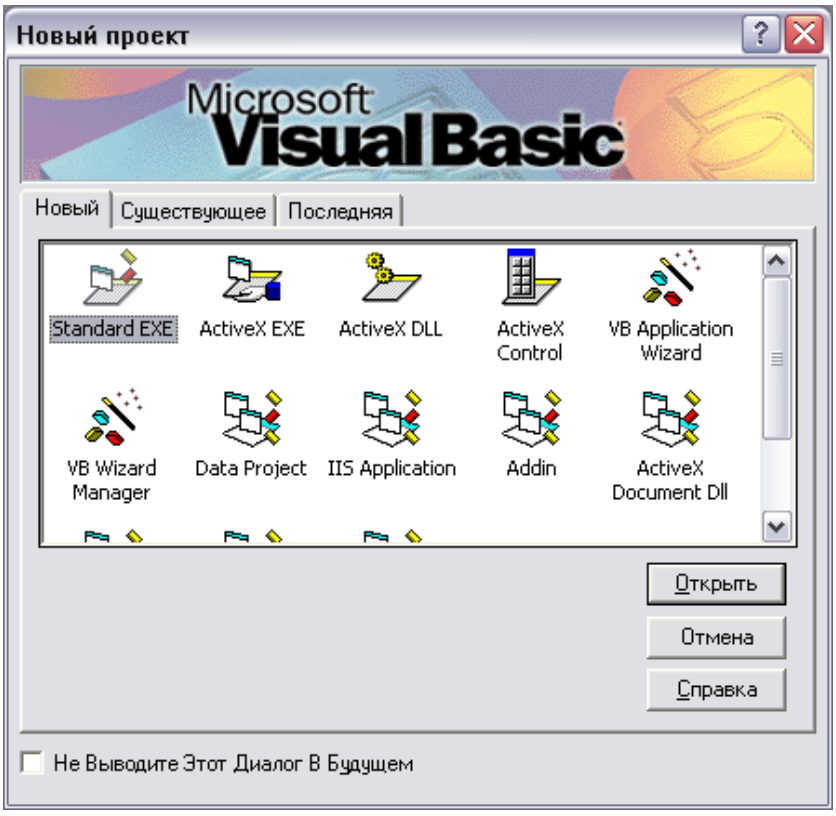



**Рисунок 2.1.** Діалогове вікно «Новый проект»

Клацнувши кнопку **Открыть**, відкриємо вікно проекту (Рис.2.2).

Це вікно Visual Basic утримує рядок меню, панелі інструментів та різноманітні вікна.

У верхній частині екрану (під рядком меню) знаходиться центр управління Visual Basic – панель інструментів **Стандартная** (**TooIBar**). ЇЇ можна настроювати, як це звичайно робиться в додатках Microsoft.

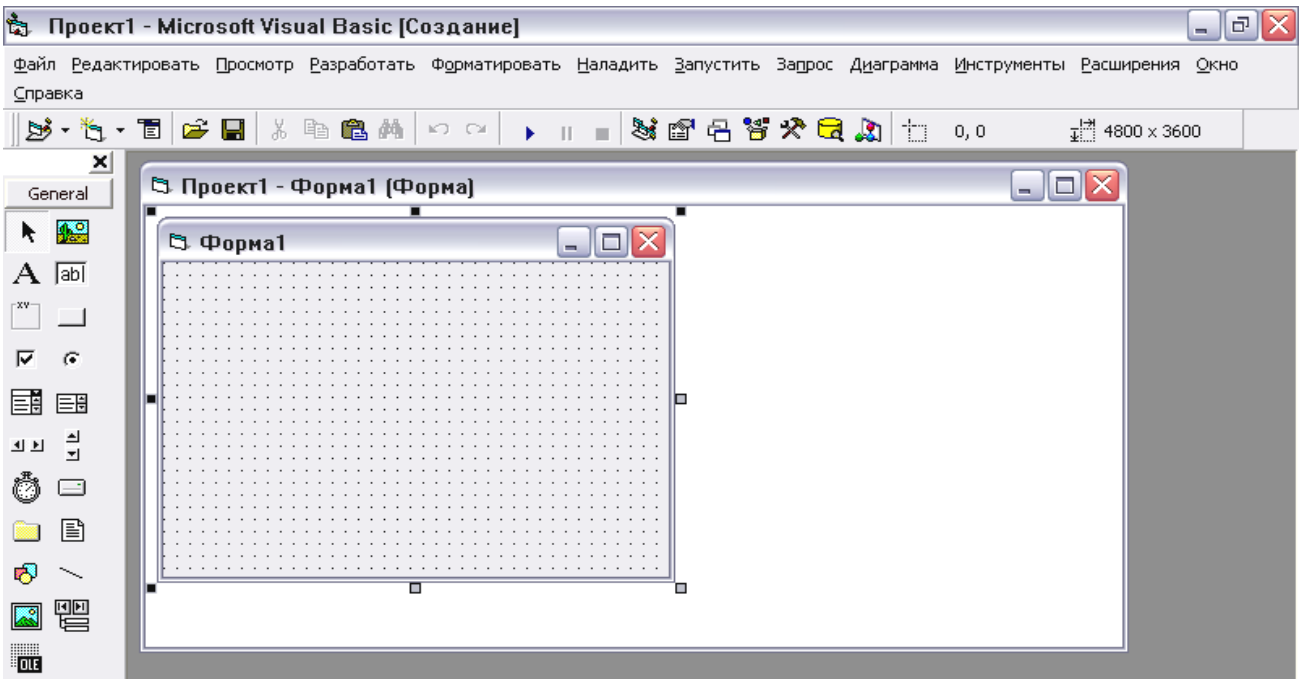



**Рисунок 2.2.** Вікно для створення нового додатку.

Кнопки, поля введення та інші елементи, що потрібні для створення додатка, розташовані на панелі елементів **TooIBox**:


**Рисунок 2.3.** Панель елементів управління «TooIbox General»

Зазвичай панель розташована ліворуч від форми. Якщо на екрані панель відсутня, натисніть кнопку **Инструменты** (**Tools**) на панелі інструментів **Стандартная**. Для вибору елемента керування потрібно клацнути на ньому і потім за допомогою миші встановити у формі його розмір і позицію. Після подвійного клацання на піктограмі елемента в центрі форми з'являється відповідний елемент стандартного розміру. Форму також можна настроювати зменшуючи або збільшуючи кількість елементів на ній. Щоб розмістити новий елемент управління на панелі, необхідно вибрати команду **Компоненты** (**Components**) у меню **Разрабо-**

**тать (Project)**. З'явиться діалогове вікно Components (Рис.2.4). Потрібно виділити прапорцем необхідний елемент і натиснути кнопку **Применить.** Обраний елемент з’явиться на панелі інструментів.

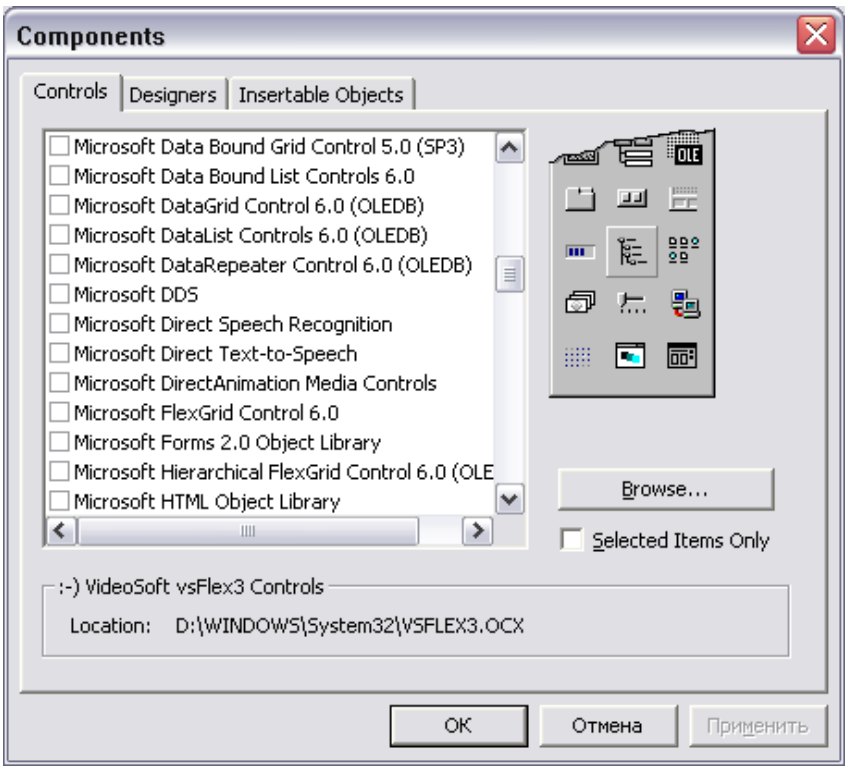



**Рисунок 2.4.** Вікно «Компоненты» для виведення нових елементів керування на панель елементів.

## 2.5. ВІКНО ФОРМИ ТА ЙОГО ВЛАСТИВОСТІ

Вікно форми (рис.2.2), яке часто називається просто «форма», є головним елементом додатка. Форма являє собою контейнер для елементів керування. Крапки сітки на формі допомагають розміщенню елементів при роботі додатка.

Клацнувши два рази по заголовку форми можна змінювати розмір вікна від стандартного до екранного розміру вікна проекту та навпаки. Можна також змінювати розмір вікна форми за допомогою трьох маркерів рамки вікна форми, перетягуючи потрібний маркер в сторону збільшення або зменшення рамки. Це важливо тому, що зміст вікна форми не підганяється автоматично до його змінених розмірів. Це може привести до того, що елемент управління після зміни розміру вікна буде знаходитись поза видимою областю і ви можете забути про нього. Не включиться також і смуга прокрутки. Також змінити висоту та ширину форми в режимі проектування можна за допомогою властивостей Height та Width, які можна відшукати у вікні Свойства. Якщо розміри форми стають більші ніж розміри вікна з формою в режимі проектування, з'являються смуги прокрутки. Змінити ро-

змір форми можна також і в режимі виконання програми, що не впливає на значення властивостей Height та Width.

При запуску форма, що відкривається на екрані, не містить елементів керування. Після клацання на піктограмі необхідного елемента керування курсор миші приймає форму хрестика. Тепер потрібно вказати у формі початковий кут елемента керування, натиснути ліву кнопку миші і, не відпускаючи її, встановити розмір елемента. Після досягнення потрібного розміру кнопка відпускається й у формі з'являється обраний елемент керування.

Кожна форма зберігається в проекті у вигляді окремого файлу. Цей файл утримує опис робочого середовища і код, який відноситься до елементів керування і форми. Форми зберігаються як звичайні текстові файли.

Переглянемо нижче деякі властивості (таблиця 1).

Для зміни заголовку форми необхідно відшукати у списку властивостей **Caption**, виділити її подвійним клацанням мишки і ввести текст заголовку.

Змінити колір фону форми можна подвійним клацанням мишки на властивості BackCo1or (відкриється для вибору палітра кольорів).

Властивості Left, Top визначають положення форми на екрані при запуску програми на виконання. Це координати лівого верхнього кута форми у твіпах (1/1440 логічного дюйма (залежить від монітора)). В режимі виконання можна переміщувати форму, як звичайне вікно форми, що не впливає на властивості Left, Top.

**Таблиця 1** Властивості форми

| **Властивість** | **Опис** |
|---|---|
| Name | Ім’я програми за замовчуванням |
| Caption | Заголовок для форми |
| Height, Width | Висота та ширина форми |
| Left, Top | Координати відносно краю екрану |
| WindowState | Розмір вікна при першому з’явленні |
| BackColor | Колір фону |
| Picture | Малюнок |
| MaxButon, Min-Button | Активні або ні кнопки управління вікном |
| ControlBox | Наявність віконного меню |
| BorderStyle | Тип границі |
| ScaleHeight, ScaleWidht | Масштаб виміру для системи координат форми |
| ScaleMode | Одиниці виміру для системи координат форми |

Деякі властивості приймають тільки дозволені значення і подвійне клацання на назві властивості змінює її значення на наступне зі списку. Іноді краще відкрити весь список і вибрати необхідне. Для цього необхідно клацнути по назві властивості, а потім по кнопці, яка з'явиться, зі спадаючим списком.

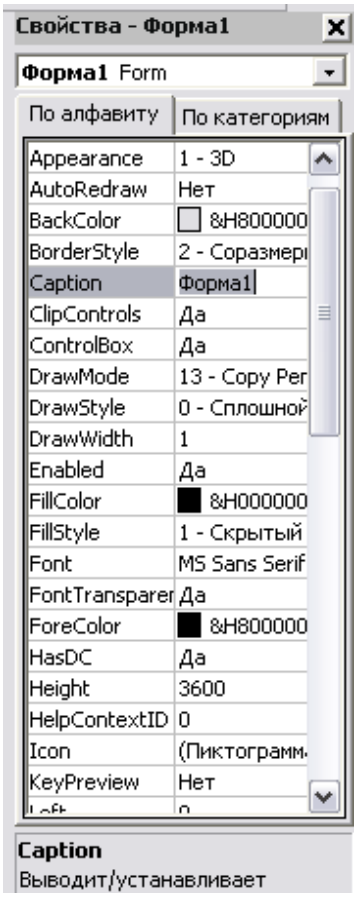



**Рисунок 2.5.**
Властивості форми

Властивість Border Style дозволяє обрати різні варіанти границь форми в режимі виконання:

0 (None) – фіксоване положення і розмір (не має рядка заголовку);

1 (Fixed Single) – фіксований розмір (не можна змінити в режимі виконання);

2 (Sizable) – всі можливості;

3 (Fixed dialog) – для вікон діалогу (можна тільки перемістити і закрити);

4 – фіксований для вікна інструментів;

5 – всі можливості для вікна інструментів.

Властивість BorderStyle має вищий пріоритет ніж властивості MaxButton. Крім стандартних властивостей Caption, BackColor, Font тощо, форми мають властивості, які притаманні тільки їм. Переглянемо нижче деякі властивості.

Властивість ControlBox визначає чи буде відображатись системне меню, за допомогою якого можна вийти з програми (**Alt-F4**). Якщо системне меню видаляється, користувач повинен передбачити інший вихід з програми.

Властивість MaxButton визначає наявність або затінення кнопки максимізації, за допомогою якої можна збільшити вікно до розмірів екрану. Якщо властивості присвоєне значення False, то відповідна кнопка буде відсутня, а команда **Maximize** (**Развернуть**) видаляється з системного меню.

Властивість MinButton визначає наявність або затінення кнопки **Свернуть окно.** Команда **Minimize** (**Свернуть** ) присутня або видаляється з системного меню.

## Основні події Форми

Необхідно відмітити одну особливість подій форми – синтаксис процедури обробки події відрізняється від синтаксису процедур обробки подій елементів управління.

*Синтаксис*

*Form_Подія ([Аргументи...])*

Ім'я процедури обробки події форми завжди утримує Form. При цьому не важливо, як фактично називається форма.

Однією з найбільш поширених подій форми є Load. Ця подія відбувається при запису форми в пам'ять. Тому найкраще Load підходить для ініціалізації об'єктів і змінних, які належать формі.

Подія UnLoad викликається, якщо форма видаляється із пам'яті. За допомогою параметру Cancel можна відмінити видалення форми з екрану.

## 2.6. ОСНОВНІ ЕЛЕМЕНТИ КЕРУВАННЯ

### Кнопка (Command Button)

*Призначення*

Цей елемент керування використовується для того, щоб почати, перервати, чи закінчити який-небудь процес. Кнопка зустрічається у всіх додатках Windows.

*Події*

Головною подією для кнопки є Click. Крім цієї події у кнопки можуть бути і інші, але вони застосовуються рідко.

Для виклику події Click є різні способи: найпростіший – безпосереднє клацання на кнопці мишею. Ця ж подія викликається також, якщо за допомогою клавіші **TaЬ** перемістити фокус на кнопку, а потім натиснути клавішу **Enter**. Можна програмно викликати подію Click, установивши значення властивості Value рівним True, що доступне тільки під час виконання.

*Властивості*

Є дві цікавих властивості кнопки, зв'язаних з подією Click. Властивість Default визначає, що дана кнопка є активною за замовчуванням. Якщо ця властивість дорівнює True, то натисканням клавіші **Enter** автоматично генерується подія Click цієї кнопки, незалежно від того, який елемент має фокус. Привласнювати значення True цій властивості можна тільки для однієї кнопки у формі. Варто враховувати, що в цьому випадку натискання клавіші **Enter** перехоплюється і передається цій кнопці. Звичайно кнопкою за замовчуванням є кнопка **Ok.**

Властивість Cancel використовується подібно Default. Воно забезпечує перехоплення клавіші **Esc** і виклик події Click для відповідної кнопки. Звичайно цю властивість мають кнопки **Cancel** (**Отмена**).

**Напис (Label))**

*Призначення*

Напис (Label) призначено для відображення тексту, який користувач не може змінити з клавіатури.

*Події*

Хоча деякі події цього елемента управління можна відображати, однак ця можливість не використовується.

*Властивості*

Найважливішою властивістю напису є Caption, яка утримує відображений текст. Скориставшись властивістю Border-Style, можна встановити спосіб відображення тексту з рамкою або без неї. Оформлювати текст можна, використовуючи всі можливості форматування тексту, які доступні у вікні властивостей (від виду і розміру шрифту до кольору символів). Якщо текст довший за поле напису, то частина тексту, яка виходить за поле, не відображається. Щоб цього не сталося, необхідно присвоїти значення True властивості AutoSize, що приведе розмір напису у відповідність з довжиною тексту. Таким же чином можна коригувати розмір напису по вертикалі. Для цього необхідно встановити властивість WordWrap. Слова, які не вміщуватимуться у рядок, автоматично будуть переноситись у наступний рядок.

**Текстове поле (TextBox)**

*Призначення*

Текстове поле (TextBox) є основним елементом управління, який призначений для введення даних.

*Події*

При використанні текстового поля викликає цікавість декілька подій. Насамперед, подія Change, яка викликається при зміні змісту текстового поля, ця подія відбувається кожен раз при введенні, знищенні або зміні символу. Наприклад, при введенні в текстове поле слова "так" подія Change відбувається три рази – по одному разу для кожної літери.

Для аналізу введеного в поле тексту найкраще всього підходить подія LostFocus. Ця подія викликається після того, як текстове поле зробиться неактивним (після передачі фокуса іншому елементу, коли користувач закінчить введення). Однак, якщо це поле є єдиним елементом управління в формі, то воно не може втратити фокус.

Для того, щоб видалити або ініціалізувати зміст текстового вікна, використовується подія GotFocus. Вона викликається тоді, коли користувач "входить" в текстове вікно.

*Властивості.*

Самою важливою властивістю є Text. Ця властивість відображає в полі текст.

Елементи управління, які дозволяють введення символів, мають властивість Text, а елементи, які призначені тільки для відображення тексту – властивість Caption. Текстове поле подібне маленькому редактору. Для використання його у цій якості достатньо встановити властивість Multiline це дає можливість вводити у поле декілька рядків.

В багаторядковому полі для переходу на новий рядок можна використовувати клавішу **Enter**. Але при цьому слід пам'ятати, що можливо, для деякої іншої кнопки. встановлена властивість Default, тому натискання клавіші **Enter** викличе спрацювання цієї кнопки. В такому випадку для переходу на новий рядок найкраще використовувати комбінацію клавіш **Ctrl**+**Enter** або **Shift**+**Enter**.

**Прапорець (CheckBox)**

*Призначення*

Прапорці – це елементи управління, які можна відмічати (ставити "галочку"), вибираючи з ряду опцій одну або декілька. CheckBox може мати два різних стани – відмічений і не відмічений. Власне, він може мати і третій стан. В цьому випадку елемент управління відображається як відмічений, але недоступний. Встановити такий стан елементу управління можна тільки програмно.

*Подія*

Найважливішою для прапорця, як і для кнопки, є подія Click.

*Властивості*

Єдиною важливою властивістю елемента управління CheckBox є його значення (Value). В залежності від того прапорець відмічений або ні, Value може приймати наступні значення:

0 – не відмічений,
1 – відмічений,
2 – відмічений але не доступний.

**Перемикач (OptionButton)**

*Призначення*

Цей елемент управління призначено для встановлення тільки однієї опції із групи. Зазвичай всі перемикачі форми об'єднані в одну групу. Якщо ви бажаєте

сформувати нову групу перемикачів, то потрібно помістити їх в окремий елемент-контейнер, наприклад Frame.

*Події*

Для перемикачів важлива тільки одна подія – Click.

*Властивості*

Важливішою властивістю перемикачів є Value. З її допомогою можна встановити стан перемикача. Ця властивість може приймати значення True або False.

**Список (ListBox)**

*Призначення*

Список ListBox дозволяє користувачу вибирати зі списку один або декілька елементів. В будь-який час в список можна додавати нові елементи або видаляти існуючі. Якщо не всі елементи можуть відображатися у полі списку, то в ньому автоматично відображається смуга прокрутки.

*Події*

Основна подія списку – Click. Ця подія викликається, якщо користувач за допомогою мишки або клавіш управління вибере елемент зі списку.

*Методи*

Вікно списку – це перший з елементів управління, для якого важливу роль грають методи. Методи списку необхідні для обробки елементів списку – додавання або видалення. Для додавання нових елементів використовується метод AddItem: ListBox.AddItem Елемент [, Індекс]

Для видалення елемента зі списку використовується метод RemoveItem, якому як параметр передається індекс елементу, який видаляється. Індексація елементів починається з нуля (0):

ListBox.RemoveItem – Індекс елемента

Для видалення усіх елементів використовується метод Clear:

ListBox.Clear

*Властивості*

Використання властивості списку Text – найпростіша можливість отримати текст вибраного елемента списку. В будь-який момент часу значення цієї властивості утримує текст вибраного елемента списку або пустий рядок, якщо жоден елемент не обраний.

Властивість Sorted визначає спосіб розташування елементів у списку. Якщо встановити цю властивість, то всі елементи у списку будуть відсортовані по алфавіту, навіть якщо вони були добавлені з зазначенням індексу. Індекс останнього добавленого елементу утримує властивість NewIndex.

Ця властивість пов'язана з іншою цікавою властивістю – ItemData(), за допомогою якого кожному елементу списку можна поставити у відповідність число типу Long. Використовуючи цю властивість, ви можете скласти, наприклад, список співробітників, зберігши їх індивідуальні номери у властивості ItemData.

**Поле зі списком (ComboBox)**

*Призначення*

Поле зі списком або ComboBox – це комбінований список, який являє собою комбінацію двох елементів управління – самого списку зі значеннями і поля введення тексту (текстового поля). Поля зі списком використовуються у тому випадку, якщо попередньо не можна визначити значення, яке потрібно включити до списку, або список утримує багато елементів. В такому списку потрібне значення можна не тільки вибирати, але і безпосередньо вводити в поле введення. Нове значення після введення автоматично поміщається у список.

*Події*

Для поля зі списком важливу роль відіграють події як поля введення, так і списку. Основні з них – Click, яка використовується для вибору елемента списку, та Change – зміна запису в полі введення тексту.

*Властивості*

Поле зі списком має майже всі властивості текстового поля TextBox і списку ListBox (виключенням є властивість MultiLine). Існує властивість Style, яка визначає зовнішній вигляд і функціонування поля зі списком.

**Смуги прокрутки (ScrollBar)**

*Призначення*

Елемент управляння ScrollBar – це смуги прокрутки вікна. Смуга прокрутки як елемент управління не виконує автоматично будь-яких дій, її поведінку необхідно програмувати. Існують горизонтальна і вертикальна смуги прокрутки.

*Події*

Мають місце дві цікаві події: Change, яка виникає після зміни позиції бігунка або після програмної зміни значення властивості Value, та Scroll, яка відбувається під час прокрутки (коли користувач схопив і переміщує бігунець).

*Властивості*

Перед використанням смуг прокрутки необхідно встановити для них діапазон, який показує кількість кроків прокрутки між крайніми позиціями бігунка.

###  Таймер (Timer)

*Призначення*

За допомогою таймера можна запускати або завершувати процеси додатків в визначені моменти часу. Таймер може бути корисним і у випадку коли додаток виконується в фоновому режимі. Під час проектування таймер відображається в формі, але під час виконання програми він є невидимим.

*Події*

Таймер має єдину подію – Timer, яка викликається по закінченню встановленого часового інтервалу.

*Властивості*

Для установки інтервалу часу існує властивість Interval (в мілісекундах). Якщо встановлена властивість Timer незалежно від того, який додаток активний, вона спрацьовує через певний час. Для відключення таймеру необхідно присвоїти властивості Interval значення 0 або властивості Enabled значення False.

###  Список пристроїв (DriveListBox)

*Призначення*

Елемент управління DriveListBox призначений для відображення і роботи з дисками, каталогами і файлами. DriveListBox відображає списки всіх доступних дисків і пристроїв системи та забезпечує можливість їх вибору.

*Події*

Самою цікавою подією елемента DriveListBox є Change. Ця подія викликається при зміні носія даних.

*Властивості.*

Елемент DriveListBox найчастіше використовує властивість Drive, яка повертає обраний диск або пристрій (наприклад, "C:\").

###  Список каталогів (DirectoryListBox)

*Призначення*

DirectoryListBox або коротко DireListBox – цей елемент управління призначений для вибору файлів. Він відображає структуру обраного диску і дозволяє здійснити вибір і зміну каталогу.

*Події*

Change – викликається в результаті подвійного клацання мишкою на імені каталогу у вікні перегляду.

*Властивості*

Головною властивістю є Path, яка повертає повний шлях до обраного каталогу (наприклад, C:\Windows\System).

**Список файлів (FileListBox)**

*Призначення*

FileListBox відображає файли поточного каталогу, звідки їх можна обирати.

*Події*

Основною подією є Click, яка викликається при виборі користувачем імені файлу в списку. Подія PathChange виникає після зміни шляху (властивість Path), а подія PatternChange виникає після зміни маски вибору файлів (властивість Pattern).

*Властивості*

Цей елемент має багато спільних властивостей з елементом ListBox. Однак основною його властивістю є властивість FileName, яка утримує ім'я обраного файлу.

**Рамка (Frame)**

*Призначення*

Рамка (Frame) – це один з елементів контейнерів. Його призначення – об’єднувати в групу декілька елементів управління. Об'єкти, які об'єднані за допомогою рамки, можна як єдине ціле переміщувати, активізувати та деактивізувати, робити видимими або невидимими.

*Події*

Події Рамки зазвичай не обробляються, хоча при необхідності це можна зробити.

**Графічний фрейм (PictureBox)**

*Призначення*

Призначений для відображення рисунків та інших графічних об'єктів Цей елемент управління також є елементом-контейнером, тому його можна використовувати для об'єднання інших елементів.

*Події*

Події PictureBox зазвичай не обробляються, хоча при необхідності можна зробити.

*Властивості*

Положення PictureBox у формі задається властивістю Align, яка визначає чи буде PictureBox у одного з країв форми чи збереже положення, задане розробником проекту. Якщо елемент управління закріплюється у одного з країв форми, то його розмір (ширина або висота) завжди встановлюється в відповідності з розміром форми.

Властивість AutoSize визначає, чи будуть автоматично змінюватись розміри елемента управління для відображення рисунків будь-якого розміру.

Сама важлива властивість PictureBox – Picture, яка утримує відображений графічний об'єкт. Це може бути растрове зображення (*.BMP), піктограма (*.ICO), мета файл (*.WMF) або розширений мета файл (*.EMF), а також GIF та JPEG-файл.

При виконанні додатків для зміни властивостей використовується функція LoadPicture:

```
Picture1.Picture = LoadPicture("C: \Windows \AuTos.Bmp")
```

Зберегти зображення можна за допомогою функції SavePicture:

```
SavePicture Picture1.Picture,"Build.BMP"
```

**Зображення (Image)**

*Призначення*

Елемент управління Image також створений для відображення рисунків. На відміну від PictureBox, він не є елементом-контейнером. Він не дозволяє малювати та не допускає групування об'єктів. Однак Image використовує менше ресурсів і перерисовує швидше, ніж PictureBox. Тому для відображення малюнків Image може бути кращим варіантом.

*Події*

Події Image зазвичай не аналізуються.

*Властивості*

Головна властивість – Picture. За її допомогою можна визначити малюнок на стадії програмування або при виконанні програми. Властивість Stretch визначає, як відображується рисунок. Якщо значення властивості Stretch =True, то розміри малюнка змінюються до розмірів елементу управління Image, в протилежному випадку елемент управління змінюється до розмірів малюнка.

**Фігура (Shape)**

Відображення геометричних фігур у формі.

**Лінія (Line)**

Зображення графічних ліній.

**Данні (Data)**

З’єднання з існуючою базою даних.


**OLE**

Додавання до додатку функцій інших програмних засобів.

## Фокус

*Фокус* – одне з найважливіших понять при зверненні до елементів управління в Windows. Основною проблемою при використанні клавіатури є те, що для обслуговування багатьох елементів управління є тільки одна клавіатура. Система Windows приймає рішення, якому додатку передати введену з клавіатури інформацію – управління отримує активний елемент, той що має фокус.

Якщо елемент отримує фокус, то це відповідним чином відображається на екрані –текстове поле відображається з мигаючим маркером введення, командна кнопка виділяється пунктирною рамкою навколо надпису. Фокус встановлюється подвійним клацанням на елементі управління.

## Вікно властивостей – Properties

У цьому вікні задаються властивості обраного елемента керування.

У рядку заголовка вікна властивостей (рис. 2.5) поруч з текстом Properties вказується ім'я форми, якій належить елемент керування. Поле зі списком під рядком заголовка дозволяє обрати необхідний елемент керування. У списку, розташованому нижче, перераховані властивості цього елемента (за абеткою або по категоріях). Набір властивостей залежить від типу елемента керування.

Список властивостей складається з двох стовпців: ліворуч перераховані назви властивостей, а праворуч – їхні значення. Редагування властивості здійснюється або вручну (наприклад, введення імені елемента), або вибором відповідного поля зі списку, або за допомогою діалогового вікна настроювання властивості. Короткий опис обраної властивості відображається в нижній частині вікна.

У цьому вікні задаються властивості обраного елемента керування.

У рядку заголовка вікна властивостей поруч з текстом Properties вказується ім'я форми, якій належить елемент керування. Поле зі списком під рядком заголовка дозволяє вибрати необхідний елемент керування

У коді програм можуть бути імена елементів керування. Ім’я слід утворювати з одного чи кількох префіксів і базового імені, тобто воно має вигляд: <*Префікс*><*Базове ім’я*>. Префікс використовується для вказування типу даних об’єкта, а базове ім’я – для його призначення. Загальноприйняті префікси для імен основних елементів керування подано у таблиці 2.

**Таблиця 2.** Загальноприйняті префікси

| Тип об’єкта | Призначення | Префікс |
|---|---|---|
| Label | Напис | lbl |
| TextBox | Текстове поле | txt |
| CommandButton | Кнопка | cmd |
| CheckBox | Прапорець | chk |
| OptionButton | Перемикач | opt |
| Frame | Рамка | fra |
| ListBox | Список | lst |
| ComboBox | Поле зі списком | cbo |
| Image | Малюнок | img |
| PictureBox | Графічний фрейм | pic |
| OLE Container | Об’єкт OLE | ole |
| Form | Форма | frm |

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Які мови називаються машинними та машинноорієнтованими?
2. Яка система числень використовується у ЕОМ?
3. Що означає термін «візуальне програмування», або «об’єктно-орієнтоване програмування»?
4. Які можливості має система програмування Visual Basic?
5. З яких компонентів складається головне вікно Visual Basic? Яке призначення вони мають?
6. Якими засобами потрібно користуватись, щоб у середовищі Visual Basic відображалися потрібні компоненти?
7. Яке призначення має панель елементів керування?
8. Що таке фокус?
9. Яке призначення має вікно властивостей? Які основні елементи керування ви знаєте?
10. Що таке інтерпретатор та компілятор? Для чого вони призначені?
11. Які дії виконуються у вікні коду?
12. Що означає, що проекти Visual Basic будуються по модульному принципу? Як це впливає на збереження проекту?
13. Яке призначення мають елементи керування та якими властивостями вони володіють?
14. Яким чином можна відкрити вікно коду та які дії у ньому виконуються?
15. Що треба зробити щоб створений вами проект можна було запускати навіть на комп’ютерах де не встановлено пакет Visual Basic?

# *Лекція* 3

# ОСНОВИ СИСТЕМИ ПРОГРАМУВАННЯ VISUAL BASIC

***План лекції:***

*3.1. Елементи системи програмування Visual Basic*

*3.2. Робота у вікні коду*

*3.3. Використання змінних у програмі*

*3.4. Константи: змінні, які не змінюються*

*3.5. Функції Visual Basic*

*3.6. Арифметичні вирази*

*3.7. Логічні вирази*

*3.8. Структура процедури Visual Basic*

*3.9. Оператори системи програмування Visual Basic*

## 3.1. ЕЛЕМЕНТИ СИСТЕМИ ПРОГРАМУВАННЯ VISUAL BASIC

Програми в Visual Basic керуються подіями. Для кожного об'єкта в системі визначений перелік подій, що його стосується. Реакцію на дії можна запрограмувати. Для цього за допомогою редактора коду створюються процедури обробки подій.

& – об'єднання рядків ;

Mod – залишок від ділення.

Операції \, Mod, & – корисні в спеціальних математичних формулах при обробці текстової інформації.

**Таблиця 3.** Елементи системи програмування

| № п/п | Символ | Пояснення |
|---|---|---|
| I | | |
| 1 | A-Z(a-z) | Букви латинського алфавіту |
| II | | |
| 1 | 0-9 | Арабські цифри від 0 до 9 |
| III Знаки арифметичних операцій | | |
| 1 | + | Плюс |
| 2 | - | Мінус |
| 3 | * | Множення |
| 4 | / | Ділення |
| 5 | \ | Цілочислове ділення |
| 6 | ^ | Піднесення до степеню |

| IV Роздільні знаки | | |
|---|---|---|
| 1 | . | Крапка |
| 2 | , | Кома |
| 3 | : | Двокрапка |
| 4 | ; | Крапка з комою |
| 5 | ‘ | Апостроф |
| 6 | " | Лапки |
| 7 | ( | Дужка, що відкривається |
| 8 | ) | Дужка, що закривається |
| V Символи для об'яви типу даних | | |
| 1 | % | Цілі |
| 2 | & | Довгі цілі |
| 3 | ! | Дійсні звичайної точності |
| 4 | # | Дійсні подвійної точності |
| 5 | $ | Символьні |
| VI Знаки відношень | | |
| 1 | = | Дорівнює |
| 2 | > | Більше |
| 3 | < | Менше |
| 4 | >= | Більше або дорівнює |
| 5 | <= | Менше або дорівнює |
| 6 | <>, >< | Не дорівнює |

## 3.2. РОБОТА У ВІКНІ КОДУ

Всі процедури обробки події в Visual Basic реалізуються у вікні коду (рис.3.1). Для відображення цього вікна необхідно виконати наступні дії:

1. відкрити нову форму Visual Basic (рис.2.2).
2. на панелі інструментів **Стандартная** натиснути кнопку **Исследователь программы**.

У вікні, що з'явиться, натиснути кнопку **Просмотреть код**. Відкриється вікно Проект1-Форма1 (Код) (рис.3.1).


**Рисунок 3.1.** Вікно коду

Блок коду, що зв'язаний з об'єктом інтерфейсу, називається процедурою події Visual Basic. Це дозволяє також відкривати вікно коду подвійним клацанням на елементі управління у формі. У верхній частині вікна є два списки вибору об'єкту (**Общее**) та події (**Описание**), на які буде реагувати програма. Обравши ім'я об'єкту та ім'я коду події, у вікні змісту одержимо "заготовку", яка складається з дужок операторів – об'яви процедури та кінця процедури. Ім'я процедури створюється з імені об'єкта та назви події. Між ними потрібно вписати команди обробки події.

**Приклад**. Після вибору об'єкта Комманда1 та події Click, відкриється вікно коду процедури *Private Sub Комманда1_Click* (рис. 3.1). Тіло процедури (програми) записується між заголовком і оператором *End Sub*.

Програма – це реалізація алгоритму в системі Visual Basic 6.0. Код програми – це послідовність команд (операторів). Команди містять ключові слова й параметри Visual Basic.

Основною структурною одиницею програми є рядок. В одному рядку може бути записано декілька команд. При цьому команди розділяються символом (:). Інколи команда доволі довга і не входить в один рядок. В цьому випадку використовується символ переносу рядка (_). Символу переносу рядка повинен передувати пропуск.

**Зауваження!** Не дозволяється розбивати переносом рядкові константи. У програмі використовуються коментарі, які пояснюють дії команд. *Коментар* – це довільний текст після знаку апострофа.

Редактор коду відслідковує зміст введеного тексту і надає допомогу: виділяє кольором ключові слова, автоматично вставляє потрібні пропуски і прописні літери.

Редактор виділить кольором і видасть повідомлення при намаганні перейти до наступного рядка, якщо поточний рядок вміщує помилки.

Якщо ввести ім'я керуючого елемента і рядом поставити крапку, то випливає список властивостей і методів елемента для вибору. Використання списку суттєво полегшує роботу. Вибір можна завершити натисканням клавіші **Enter**. При цьому обране слово додається до тексту, курсор переміщується на наступний рядок або **Ctrl+Enter**, тоді курсор залишається в тому ж рядку.

Виділені фрагменти тексту можна переносити і копіювати. Доступні всі операції з буфером обміну, вставка тексту з файлу і інші команди меню **Правка**. Можлива настройка редактора в діалозі **Options** на вкладці **Editor** меню **Tool**.

Після запуску процедури на виконання (кнопка **Начать** на панелі **Стандартная**) отримаємо результати.

**Вихід з програми**

Оператор **End** негайно закриває додаток. Після оператору **End** не виконується жоден код і не відбувається жодна подія.

## 3.3. ВИКОРИСТАННЯ ЗМІННИХ У ПРОГРАМІ

*Змінна* – іменоване місце в пам'яті комп'ютера, яке має ім'я (ідентифікатор) та значення.

*Ідентифікатор* – це послідовність букв, цифр та спеціальних символів (!. $, @, %, &), що починається з букви. Довжина ідентифікатора ≤ 20 символів. Ім'я не повинно вміщувати крапку, співпадати з ключовим словом Visual Basic, повинно бути унікальним в межах області видимості.

Інколи необхідно зберігати деякі значення під час виконання програми. Для цього змінні різних типів оголошуються за допомогою оператора об'яви змінних.

**Формат:**

Повний синтаксис оголошення змінних:

*Public / Private / Dim <ім'я змінної> [As <ім'я типу>]*

У квадратні дужки узята необов'язкова частина команди. Знак / замінює слово "або".

Перше ключове слово визначає область видимості змінної:

*Public* – глобальне (відкрите), доступне всьому додатку;

*Private* – доступна на рівні модуля форми;

*Dim* – змінна доступна на рівні модуля форми або тієї процедури, де була об'явлена.

### ОБЛАСТЬ ВИЗНАЧЕННЯ ЗМІННИХ

Дуже важливою характеристикою змінних є область їх визначення. В Visual Basic є три види областей визначення, які характеризують доступність змінної.

**Локальні змінні**, які визначаються всередині процедури або функції. Вони доступні тільки всередині цієї процедури. Локально об'явлені змінні при виході з процедури вилучаються з пам'яті, та при новому виклику процедури ініціалізуються заново, їх зміст при цьому не зберігається, що не завжди доцільно. Локальні змінні потрібно описати всередині цієї процедури з ключовим словом **Dim.**

**Змінні контейнера** визначаються в секції (General) (Declarations) і доступні тільки всередині відповідного контейнера (форми, модуля або класу).

**Глобальні змінні** визначаються в секції (General) (Declarations) модуля. При цьому замість оператора **Dim** використовується зарезервоване слово **Public**. Глобальні змінні доступні у всіх модулях і процедурах проекту. При запобіганні помилок в програмі і для підвищення її ефективності всі змінні, що використовуються в програмі, повинні бути оголошені із зазначенням типу даних (табл. 4). При оголошенні змінних їм відводиться місце в пам'яті, визначається спосіб зберігання. Тип змінної також може бути визначений за допомогою суфікса.

## Типи даних

**Таблиця** 4. Типи даних Visual Basic

| Тип даних | Розмір | Діапазон значень | Префікс | Символ |
|---|---|---|---|---|
| Integer (ціле) | 2 байта | Від -32 768 до 32 767 | Int | % |
| Long Integer (Довге ціле) | 4 байта | Від -2147483648 до 2147483647 | Lng | & |
| Single-precision Floating point (Одинарної точності з плаваючою десятковою крап- | 4 байта | Від -3.402823 E 38 до 3.402823 E 38 | Sng | I |
| Double precision Floating point (Подвійної точності з плаваючою десятковою крапкою) | 8 байт | Від 1.797693134862320E308 до 1.797693134862320E308 | Dbl | # |
| Currency (Грошові одиниці) | 8 байт | Від-922337203685477.5808 | Cur | @ |
| String (рядок) | 1 байт на символ | Від 0 до 65 535 символів | Str | $ |
| Boolean | 2 | True False | Bln | |
| Date (Дата) | 16 байт | Від 1.01. 100 до 31. 12.999 | Dtm | |
| Variant (Варіант) | (для чисел) 22 байта + 1 байт на символ | Для всіх типів даних (встановлюється по замовчуванню в операторах описання) | Vnt(var) | |

**Таблиця 5.** Приклади оголошення змінних

| Тип даних | За допомогою типа даних | За допомогою суфікса |
|---|---|---|
| Ціле | Dim x AS Integer | Dim x % |
| Дійсне одинарної точності | Dim x As Single | Dim x! |
| Дійсне подвійної точності з плаваючою десятковою комою | Dim x AS Double | Dim x # |
| Варіант | Dim x | |

**Зауваження!** Змінну можна оголосити без використання оператора **Dim** (неявний опис) за допомогою оператора присвоювання: x =- 5.6; y="Іванов".

Якщо в подальшому ім'я змінної буде введено невірно, то Visual Basic не виявить помилки.

**Приклад.** Виконаємо процедуру, замінюючи в ній опис даних.

1) Dim x AS Double, y AS Double ' Дійсне подвійної точності з плаваючою десятковою комою

x=1.78631598770806 y=1.959426478990327

2) Dim x!, y! ' Одинарної точності з плаваючою десятковою комою

x= 1.786316 y= 1.959427

3) Dim x@, y@ ' Грошові одиниці

x=1.79 y=1.96

4) Dim x AS Integer , y AS Integer ' Тип даних ціле

x=2 y=2

## 3.4. КОНСТАНТИ: ЗМІННІ, ЯКІ НЕ ЗМІНЮЮТЬСЯ

Якщо в програмі використовується змінна, значення якої ніколи не змінюється, то краще використовувати замість змінної константу. Константа являє собою умовне ім'я, що використовується замість числа, або текстового рядка, що не підлягає зміні. Дія константи схожа з дією змінної, але її значення не може бути змінено в процесі використання програми. Константи оголошуються за допомогою ключового слова **Const.**

Повний синтаксис оголошення:

*Public/ Private I Const < ім'я > [ AS < ім'я типу > | = < значення >*

**Приклад**.

Const Pi=3.14159

Щоб константа була доступна всім процедурам програми, необхідно створити її в стандартному модулі, записавши ключове слово **Public** перед описанням константи.

**Приклад**.

Public Const As Single Pi=3.142

## 3.5. ФУНКЦІЇ VISUAL BASIC

### СТАНДАРТНІ ТИПИ ФУНКЦІЙ

*Функція* – це оператор, що виконує певні дії та повертає результат роботи у програму. Функція може мати один або кілька аргументів, які беруться в дужки і відокремлюються між собою комами.

Стандартними стосовно Visual Basic 6.0 називаються такі функції, обчислення яких є складовими компонентами системи Visual Basic.

По призначенню вбудовані функції поєднуються в наступні групи:

- фінансово-математичні функції;
- функції перетворення типу;
- математичні функції;
- функції статуса;
- функції обробки рядків;
- функції дати та часу;
- функції для роботи з масивами;
- функції для роботи з файлами.

**Математичні функції**

В мові Visual Basic для вирішення різних математичних задач існують вбудовані функції, які залежать від одного аргументу та які можна використовувати безпосередньо при обчисленнях будь-яких виразів (табл. 6).

**Таблиця 6.** Перелік основних математичних функцій

| **Математичний запис** | **Запис на мові Visual Basic** | **Призначення** |
|---|---|---|
| \|x\| | Abs(x) | Модуль |
| arctgx | Atn(x) | Арктангенс |
| cosx | Cos(x) | Косинус |
| sin x | Sin(x) | Сінус |
| tgx | Tan(x) | Тангенс |
| $e^x$ | Exp(x) | Експонента |
| lnx | Log (x) | Натуральний логарифм |
| $\sqrt{x}$ | Sqr(x) | Квадратний корінь |

Звертаємо увагу, що всі тригонометричні функції використовують аргумент, який виражений в радіанах. Щоб перейти від градусів до радіан використовуємо формулу: αрад=α π/180.

**Таблиця 7.** Математичні функції, які можуть бути отримані із вбудованих математичних функцій Visual Basic

| Математичний запис | Запис на visual basic | Пояснення |
|---|---|---|
| sec x | 1/Cos(x) | Функція секанса, аргумент в радіанах |
| cosec x | 1/Sin(x) | Функція косеканса, аргумент в радіанах |
| ctg x | 1/Tan(x) | Функція котангенса, аргумент в радіанах |
| arcsinx | Atn(x/Sqr(l-x^A2)) | Функція арксинуса, аргумент в радіанах |
| arccosx | Atn(Sqr(l-x^A2)/x) | Функція арккосинуса, аргумент в |
| $\log_a x$ | Log(x) / Log(a) | Логарифм за основою а |
| lgx | Log(x)/Log(10) | Функція десяткового логарифму |

**Рядкові функції**

Функції обробки рядків Visual Basic наведені в табл.8. Вони служать для виконання операцій з рядками.

**Таблиця 8.** Функції обробки рядків

| Ім'я функції, параметри | Повертаєме значення |
|---|---|
| Asc (Рядковий вираз) | Повертає ASCII-код першого символа в рядковому виразі |
| Chr (Код символа) | Повертає символ, відповідний вказаному ASCII-коду |
| Instr (Початкова позиція, рядковий ви-раз_1, рядковий вираз_2) | Повертає номер позиції першого виявлення рядка 2 в рядку 1. Початкова позиція встановлює позицію рядка, з якої починається пошук (якщо параметр опущено, то пошук починається з позиції першого символа) |
| Lcase (Рядковий вираз) | Повертає рядок подібний рядку параметра, але складений із рядкових літер |
| Left (Рядковий вираз, Кількість символів) | Повертає рядок, вилучений із рядка-параметра, починаючи із початку рядка та який містить задану кількість символів |
| Len (Рядковий вираз/Ім'я змінної) | Повертає кількість символів в рядку або кількість байтів, необхідних для збереження змінної |
| LTrim (Строковий вираз) | Вилучає початкові пробіли з рядка-параметра |
| Mid (Рядковий вираз, Початкове значення [Довжина]) | Повертає рядок частку рядка (підрядок), який містить кількість символів, равноє значенню параметра Довжина, починаючи від символа із позиції Початкове значення |
| Right (Рядковий вираз, Довжина) | Повертає рядок із вказаним в параметрі Довжина кількістю символів, починаючи від кінця рядка, який задан в параметрі Рядковий вираз |
| RTrim (Рядковий вираз) | Повертає рядок, ідентичний рядку-параметра, але з вилученими кінцевими пробілами |
| Space (Довжина) | Повертає рядок пробілів вказанної довжини |
| Str (Числовий Вираз) | Повертає рядкове представлення числа |
| Val (Рядковий вираз) | Переводить рядкове представлення числа в число |

До даних типу дата/час можна застосовувати операції додавання і віднімання. Наприклад, результатом обчислення виразу Date-1 буде вчорашня дата.

До даних типу Date можуть застосовуватися ряд вмонтованих функцій, частину з яких подано у табл. 9.

**Таблиця 9.** Деякі функції для роботи з датами

| Функція | Призначення |
|---|---|
| Now | Поточні дата і час у комп'ютері |
| Date | Поточна дата у комп'ютері |
| Year (дата) | Рік в аргументі "дата" |
| Month (дата) | Місяць в аргументі "дата" |
| Day (дата) | День в аргументі дата |
| WeekDay (дата) | Номер для тижня в аргументі "дата" (неділі відповідає 1, а суботі – 7) |
| DateAdd (інтервал, кількість, дата) | Нова дата, отримана додаванням до заданої дати кількості інтервалів часу |
| DateDiff (інтервал, дата1, дата2) | Кількість інтервалів часу між першою і другою датами |

**Примітка!** У функціях DateAdd і DateDiff інтервал часу задається рядковим виразом і може набувати значень, наведених у табл. 10.

**Таблиця 10.** Припустимі значення аргументу «Інтервал»

| Значення | Опис | Значення | Опис |
|---|---|---|---|
| yyyy | Рік | w | День тижня |
| q | Квартал | ww | Тиждень |
| m | Місяць | h | Години |
| y | День року | n | Хвилини |
| d | День місяця | s | Секунди |

Наприклад, за допомогою функції DateAdd ("m", 3, Date) можна визначити дату через три місяці, а за допомогою функції DateDiff ("ww", #l,01.2001#, Date) – скільки тижнів пройшло з початку третього тисячоліття.

Значення текстового поля на формі має рядковий тип. Текстові поля використовуються для введення даних (зокрема, числових), що надалі можуть брати участь у ході виконання програми в різних обчисленнях. Щоб уникнути помилок обчислень, необхідно введені дані перетворити до числового типу. Для цього в VB є кілька функцій, поданих у табл. 11 (де x – вираз).

**Таблиця 11.** Функції перетворення типів

| Функція | Тип результату | Функція | Тип результату |
|---|---|---|---|
| CBool (x) | Boolean | CInt (x) | Integer |
| CByte (x) | Byte | CLng(x) | Long |
| CCur (x) | Currency | CSng(x) | Single |
| CDate (x) | Date | CVar(x) | Variant |
| CDbl (x) | Double | CStr(x) | String |

**Таблиця 12.** Деякі інші функції Visual Basic

| Функция | Призначення | Пояснення |
|---|---|---|
| Fix(x) | Повертають цілу частину числа, відрізняються за засобом округлення від'ємних чисел | Повертає найближче ціле число, ≥X |
| Int(x) | | Повертає найближче ціле число, ≤X |
| Rnd(x) | Функція випадкових чисел, що генерує послідовність випадкових чисел в інтервалі (0,1) | При однакових початкових значеннях генерується одна і та сама послідовність чисел. Щоб отримати при кожному новому запуску програми іншу послідовність, можна використовувати оператор Randomize |
| Round(x,[a]) | Повертає число, закруглене до заданого числа десяткових знаків | x – Обов'язковий. Число (вираз), яке заокруглюється, а -Необов'язковий. Кількість знаків після коми. Якщо параметр відсутній, функція повертає цілочислове значення. |
| Sgn(x) | Повертає знак числа | -1, x<0,<br>0, x=0,<br>1, x>0 |
| Str(x) | Перетворює числове значення в рядок | При перетворенні додатних чисел перед першою цифрою розміщується пропуск, перед від'ємними аргументами – символ "мінус" |
| Val(x) | Перетворює рядок в числове значення | x – рядок, який необхідно перетворити в число. Функція повертає значення типу Double. |

Дамо додаткове пояснення до функції Rnd(x). Аргументом x при зверненні до функції Rnd(x) може бути довільне число. Для одержання довільного цілого числа в діапазоні [A, B] можна скористатись формулою:

*Int((B-A+l)*Rnd(x))+A*

Тобто, для одержання цілого числа в діапазоні [0, 9] можна використати формулу *Int(10*Rnd(x))*.

Аргумент x можна опустити, тобто записати як *Int(10*Rnd)*.

## 3.6. АРИФМЕТИЧНІ ВИРАЗИ

*Арифметичні вирази* – це числа, змінні, функції з'єднані між собою знаками арифметичних операцій та (при необхідності) круглими дужками. Результатом арифметичного виразу є число. Всі операції з арифметичними виразами виконуються зліва направо з врахуванням пріоритету операцій та круглих дужок.

Встановлено наступний пріоритет операцій:

1. вираз в дужках ( );
2. піднесення до степеню (^);
3. присвоєння числу від'ємних значень;
4. множення та ділення (*, / );
5. ціле від ділення ( \ );
6. залишок від ділення (Mod);
7. додавання та віднімання (+,-).

Приклади друку виразів наведено у табл. 13.

**Таблиця 13.** Приклади друку виразів

| **Математичний запис** | **Запис на visual basic** |
|---|---|
| $\frac{a-b}{a+b}$ | (a-b)/(a+b) |
| $\frac{\sqrt[3]{\lvert\alpha-\beta\rvert}+arctg^3x^2}{\sqrt{x-y}+10{,}5*10^2a}$ | (Abs(Al-Be)^(1/3)+ (Atn(X^2))^3)/ (Sqr(X-Y)+10.5E2*A) |
| $\frac{sin^2x^3+l{,}5-2^5y}{ln\lvert x-y\rvert}$ | ( Sin ( x ^ 3 ))^2 + 1 . 5 - 2 ^ 5 * y ) / Log ( Abs ( x - y)) |

## 3.7 ЛОГІЧНІ ВИРАЗИ

*Логічний вираз* складається з констант, змінних, функцій, які з'єднані між собою знаками арифметичних операцій, знаками відношень та логічними операціями.

В Visual Basic існують наступні логічні операції:

Not – заперечення ,(¬)

And – кон'юнкція (логічне множення), (v)

Or – диз'юнкція (логічне додавання), (Λ)

Xor – виключення ("строга" диз'юнкція),( $\oplus$ )

Eor – еквіваленція, A $\Leftrightarrow$ B

Imp – імплікація, A→B

Логічні вирази, в яких відсутні знаки логічних операцій, називаються *простими*, а логічні вирази, в яких вони присутні, називаються *складними*. Результатом логічного виразу є два значення: True (Істина) – 1, або False (Хибність) – 0.

Результати логічних операцій наведено у таблиці 14.

**Таблиця 14.** Результати логічних операцій

| **Значення операндів** | | **Результати операцій** | | | | | |
|---|---|---|---|---|---|---|---|
| x | y | Not x | x And y y | x Or y y | x Xor y | x Eor y | x Jmp y |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

Пріоритет виконання дій в логічних виразах:

1) обчислюються арифметичні вирази (якщо вони є);
2) виконуються операції відношень;
3) виконуються логічні операції в послідовності:
   - Not
   - And
   - Or, Xor
   - Imp
   - Eor

Для зміни ходу операцій застосовуються круглі дужки. Приклади логічних виразів приведемо в таблиці 15.

**Таблиця 15.** Приклади друку логічних виразів

| **Математичне визначення** | **Логічний вираз на Visual Basic** |
|---|---|
| x Є [a,b] | x >= a and x <= b |
| x Є (∞,a] Λ x Є [ b, ∞] | x <= a Or x >= b |
| y+x v $x^2$ –y Λ x+y >B | y+x And x^2-y Or x+y > b |

**Приклад.** *Обчислити значення логічного виразу*

$(x > y)$ v $x$-$y$ > 2 Λ $x$-0,5<4, де $x$=1,5; $y$=0,6.

Логічний вираз на Visual Basic записується так:

Not($x$>$y$) And $x$-$y$>2 Or $x$- 0.5<4

Послідовність дій:

1) x > y → True (1):

2) x-y = 1,5 - 0,6 = 0,9 => 0,9 >2 → False (0);

3) x-0,5= 1,5-0.5=1 => 1 <4 →True(1);

4) Not (True) → False (0);

5) False (0)  And  False (0) = False (0);

6) False (0) Or  True (1) = True (1).

Відповідь: в результаті виконання логічного виразу одержали значення True (1).

Логічні вирази застосовуються в операторах циклів While-Wend, Do-Loop та конструкціях if, Elseif та Select Case.

## 3.8. СТРУКТУРА ПРОЦЕДУРИ VISUAL BASIC

```
Private Sub Form_<Подія> ([Аргументи])
Оператор 1
Оператор 2
……..
Оператор n
End Sub
```

Програмний оператор являє собою особливу інструкцію, яка сприймається компілятором Visual Basic.

## 3.9. ОПЕРАТОРИ СИСТЕМИ ПРОГРАМУВАННЯ VISUAL BASIC

**Оператор вводу InputBox**

*Синтаксис:*

*InputBox (<повідомлення> ,[<заголовок>], [<значення>], [<x0, y0>])*,

де <повідомлення> та <заголовок> – довільна послідовність символів в подвійних лапках ( "),

<значення> – значення за замовчуванням змінної відповідного типу;

<x0,y0> – координати лівого верхнього кутка вікна вводу на екрані;

Якщо будь-який середній елемент оператора замовчується, ставляться коми.

**Приклад.** *x = InputBox ( " x = " ,"Ввести", , 4000, 400 )*

В даному прикладі відсутнє значення за замовчуванням.

**Зауваження!** При відсутності координат вікна, вікно розташовується в центрі екрану.

При виконанні оператора з'явиться вікно в поле вводу якого слід ввести значення змінної відповідного типу. Якщо в операторі InputBox таке значення стоїть за замовчуванням, натиснути **ОК**. Введене значення присвоюється змінній і управління буде передано наступному оператору (при натисканні **Отмена** (Cansel) дія відміняється).

**Оператор виведення даних MsgBox**

*Синтаксис:*

*MsgBox (<список елементів виводу>)*

Елементами виводу можуть бути:

1.Довільна послідовність символів в подвійних лапках (у вікні оператора виведення без змін).

2. Функція Str(a) де a – аргумент.

Аргументом функції Str може бути:

a. -змінна

b. арифметичний вираз.

У випадку (а) в вікні оператора виводиться значення змінної. В випадку (б) обчислюється значення арифметичного виразу, яке виводиться у вікні оператора.

Елементи списку виводу з'єднуються між собою знаком + або &.

**Приклад**. *Вивести на екран значення змінної x та x+sinx.*

Варіант 1. *MsgBox ( " x = " +Str (x) + " " + "x+sin(x) =" + Str(x+Sin(x))*

Варіант 2. *MsgBox ( " x = " & Str (x) &" " & " x + 8in(x) = " & Str ( x + Sin(x)))*

Рядок з кодом в програмі Visual Basic називається *програмним оператором*;

Програмний оператор може бути будь-якою комбінацією ключових слів Visual Basic, властивостей, функцій, операцій та символів, сукупність яких являє собою коректну інструкцію, що розпізнає компілятор Visual Basic .

Правила, що використовуються при побудові програмних операторів називають *синтаксисом*.

**Оператор виведення даних Print (функції TaЬ, Spc, String$)**

*Синтаксис*:

*[Form n]. Print [ Список елементів виведення ],*

де до списку елементів виведення можуть входити:

• константа;

• змінні (числові чи рядкові);

• вирази;

• послідовність символів у лапках;

• функції TaЬ, Spc, String$.

Form n – виведення у форму з номером n.

*Дія*: Print – виведення результатів у форму.

Елементи списку виведення розділяються між собою комою (,) або крапкою з комою (;). В випадку розділення комою (,) наступна інформація друку-

ється з нової зони (одна зона має 14 позицій). У випадку розділення крапкою з комою (;) наступна інформація друкується безпосередньо після попередньої.

Якщо елементом списку виведення є вираз, то Visual Basic спочатку обчислює значення виразу і друкує результат. Список елементів виведення може бути і відсутнім. В такому випадку оператори виводять пустий рядок.

*Синтаксис функції* **TaЬ**:

*TaЬ(n),*

де n – номер позиції з якої буде відбуватися друк.

За допомогою операторів Print виводяться результати обчислень у вигляді десяткового числа цілого або з фіксованою комою, якщо результати знаходяться в інтервалі (0.01; 999999). В інших випадках виводяться результати з плаваючою комою.

*Синтаксис функції* **Spc**:

*Spc(n),*

де n – кількість позицій, які пропускаються при виконанні оператора Print.

*Синтаксис функції* **String$**:

*String(n," символ"),*

де n – кількість символів, які додаються при виконанні оператора Print.

Оператор Print виводить інформацію тільки у форму.

**Приклад.** *Обчислити та вивести інформацію за допомогою оператора Print.*

$$z = \frac{\sqrt{\left|x^2 - y^2\right|}}{tg^2 x} \qquad p = \frac{\sqrt[3]{y^2 - x^2} + \ln\left|x / y\right|}{\cos^2 z},$$

де x=1.5; y=-5.

Для використання оператора Print (виведення результатів у форму) необхідно виконати наступні дії:

1. Відкрити новий проект.

2. За допомогою кнопки **Command Button**, яка знаходиться на панелі елементів **General** необхідно створити командну кнопку в формі. Для цього необхідно:

а) вибрати управляючий елемент Command Button (клацнути на кнопці);

б) клацнувши лівою кнопкою миші у формі і утримуючи її натиснутою, прокреслити в формі прямокутник;

в) відпустити кнопку миші.

У формі з’явиться командна кнопка з назвою за замовчуванням Команда 1.

3. Щоб змінити назву кнопки, необхідно відкрити вікно її властивостей. Можна клацнути кнопку правою мишкою і в контекстному меню обрати пункт **Свойства**, або, клацнувши кнопку лівою мишею, на панелі інструментів **Станда-**

**ртная** натиснути кнопку **Свойства**. У вікні що виникне значення властивості **Caption** (**Надпись**) для командної кнопки Команда 1 і назву кнопки (властивість Name) замінимо на Пуск .

4. Необхідно задати певні властивості об'єкту Форма 1 (Form1). Для цього необхідно відкрити список перерахування об'єктів у верхній частi вікна Properties (Свойства) і клацнути по імені об'єкта Форма 1 (Form1). У вікні Properties (Свойства) з'являться властивості форми. Можна змінити назву форми (за бажанням) на більш конкретну (аналогічно до зміни назви командної кнопки).

5. Якщо виникне потреба змінити властивості шрифту в формі, слід у вікні Properties (Свойства) в списку по алфавіту обрати властивість Font, клацнути її, щоб відкрити вікно Вибір шрифту, і виконати необхідні зміни.

6. Обов'язково встановити для властивості Autoredraw значення Да (True).

7. Два рази клацнути по кнопці Пуск в формі. У вікні Код з'явиться заголовок процедури та його кінець:

```
Private Sub Пуск_Click()

End Sub.
```

8. Після заголовку процедури набрати текст (тіло програми).

9. Запустити програму на виконання, натиснувши кнопку **Начать** на панелі **Стандартная**.

10. У вікні Форма1 клацнути по командній кнопці Пуск. Ввести початкові значення *x* та *z* в вікні з повідомленням. Результат виконання процедури буде показаний у вікні форми (рис. 3.2).

```
Private Sub Пуск_Click()
Dim x!, y!, z!, p!
    x = 1.5: y = -5
    z = sqr(abs(x ^ 2 - y ^ 2)) / Tan(x)^2
    p = ((y^2-x^2) ^ (1 / 3) + Log(Abs(x /y))) / Cos(z) ^ 2
Print "x="; x, "y="; y
Print
Print "z="; z, "p="; p
End Sub
```

Зверніть увагу, що виведення початкових даних та результатів обчислень *z* і *p* оператором Print здійснюється безпосередньо на форму

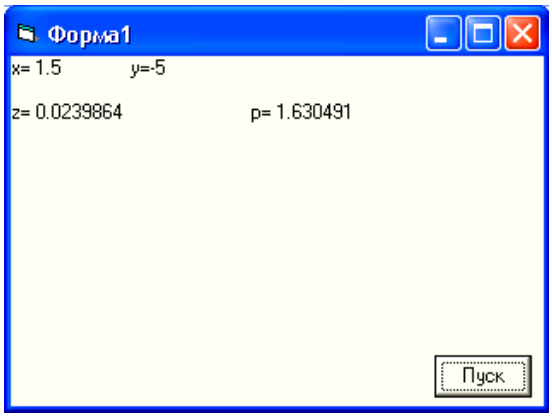



**Рисунок 3.2.** Вікно форми з результатами обчислень

11. Якщо повторно клацнути по командній кнопці Пуск, то і результат з'явиться повторно.

12. Для закінчення розрахунків необхідно клацнути по кнопці **Конец** на панелі інструментів.

13. Зберегти проект **Сохранить** (**Save Project**) або **Сохранить как** (**Save As Project**).

**Оператор присвоювання**

*Синтаксис:*

*X=A,*

де X – ім'я змінної;

A – арифметичний вираз.

= – операція присвоювання

Запис X=X+2 суперечливий з точки зору математики, в інформатиці означає, що нове значення змінної на 2 більше попереднього. Попереднє значення записується в правій частині.

Приклади друку операторів присвоювання наведено в таблиці 16.

**Таблиця 16.** Приклади друку операторів присвоювання

| **Математичний запис** | **Запис на Visual Basic** |
|---|---|
| $y = 3\frac{arctgx^2 + \cos^3 x}{\sqrt[3]{\lvert x-a \rvert}}$ | Y=3*(Atn(x^2)+Cos(x)^3)/abs(x-a)^(1/3) |
| $a = \frac{sqr(x\text{^}2 + y\text{^}2) - 1.5*10^2 x}{\cos^2 x^3 + \sin^2 y^3}$ | a=(Sqr(x^2+y^2)-1.5E2*x)/(cos(x^3)^2+sin(y^3)^2) |

**Приклад**. *Процедура з використанням операторів описання, вводу, виводу та присвоювання. Обчислити вираз*

$Y=x^2+sinz+tg^2(z+x)$, *де* $x=3.4;\ z=5.6$ .

1. Відкрити нову форму Visual Basic (рис. 2.2)

2. На панелі інструментів **Стандартная** натиснути кнопку **Исследователь Программы**.

3. У вікні **Программы** – Проект 1 (рис. 2.2) натиснути кнопку **Просмотреть код**.

4. Відкриється вікно Проект1 – Форма1 (Код).

Після вибору ім'я об'єкта Form та ім'я події Load, відкриється вікно коду процедури Form_ Load()

5. Тіло процедури (програми) записуємо між операторами Private Sub Form_Load() та End Sub.

```
Private Sub Form1_Load()
Dim x!, y!, z!
    x=Val(InputBox("x="))
    z=Val(InputBox("x="))
    y=x^2+sin(z)+tan(z+x)^2
Print x, y, z
End sub
```

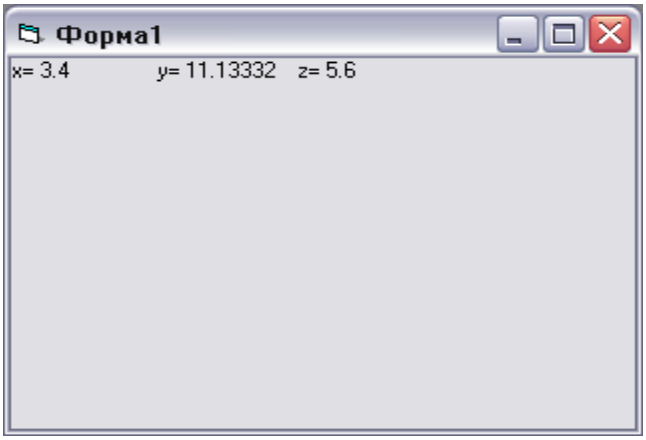



**Рисунок 3.3.**

6. Натиснути кнопку **Начать** на панелі **Стандартная**.

7. З'явиться вікно з повідомленням *x*=. В поле вводу ввести значення змінної – 3.4 і натиснути **Ok** або **Enter**.

8. З'явиться нове вікно з повідомленням *z*=. В поле вводу ввести значення змінної – 5.6 і натиснути **Ok** або **Enter**. З'являться результати обчислень:

x=3.4 z=5.6 y =11.13332.

Для введення і виведення інформації (початкових даних, коментарів, пояснень, результатів ) можна також використовувати такі елементи управління як TextBox, ComboBox, ListBox, FlexGrid.

**Приклад**. *Використаємо елементи TextBox щоб створити інтерфейс додатка для обчислення накопичень на внески, блок-схема алгоритму якого було розглянуто раніше (Приклад 1).*

1. Відкриємо нову форму.

2. Встановимо на ній 4 елементи TextBox . За замовчуванням VB дасть їм назви відповідно Текст1, Текст2, Текст3, Текст4. Погодимось з такими іменами, але очистимо поля елементів, підготувавши їх до введення даних. Для цього клацнемо правою мишкою на елементі Текст1 і в контекстному меню його клацнемо лівою мишкою пункт **Свойства**. Праворуч від форми або над нею з'явиться вікно **Свойства–Текст1**.

3. Виділимо властивість Text і видалимо слово" Текст1". Воно щезне і з поля елемента Текст 1. Якщо не закривати вікно Свойства –Текст1 і клацнути на елементі Текст2, то панель Свойства –Текст1 автоматично буде заміщена панеллю Свойства-Текст 2, навіть з виділеною властивістю Text. Очистимо її. Так само очистимо поля інших елементів TextBox.

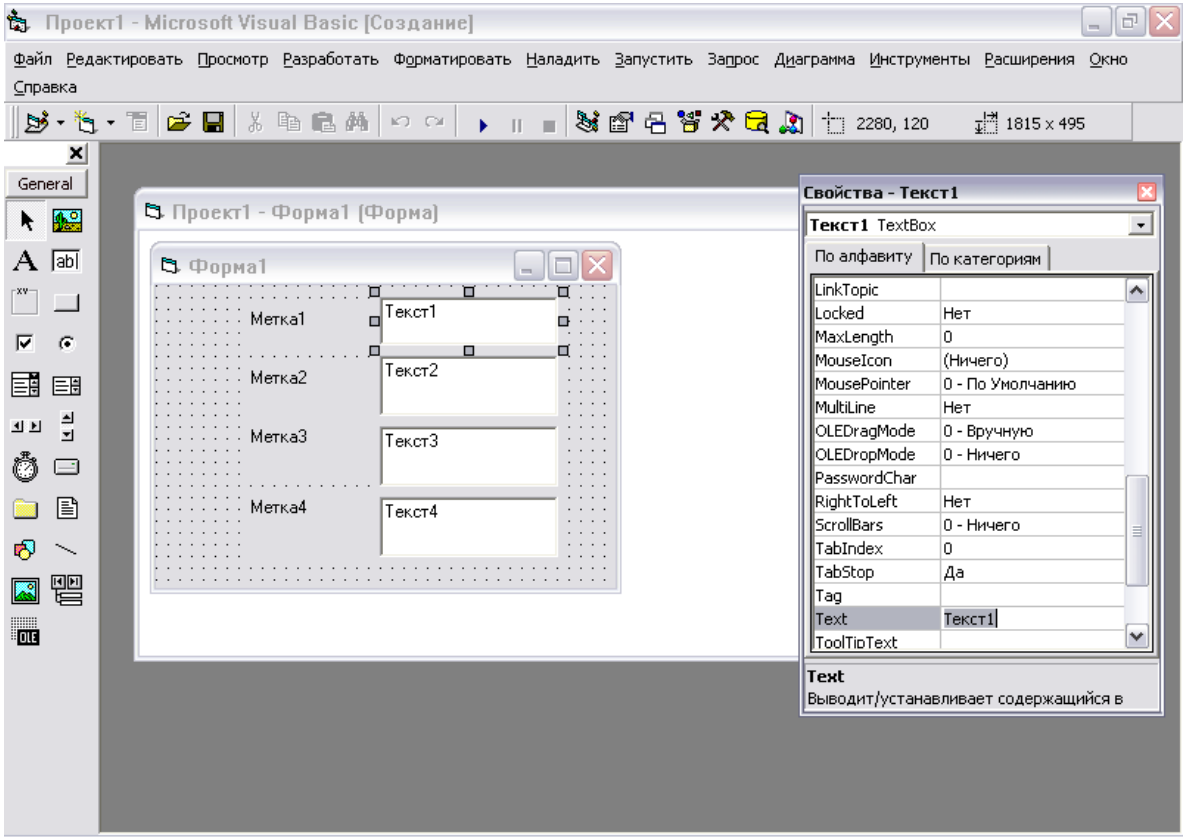



**Рисунок 3.4.**

4. Ліворуч від текстових вікон встановимо елементи Label (Мітка). Відкриємо панель Свойства елемента Метка1. Властивість Caption (Надпис) «Метка1» змінимо на «Внесок (грн.)». Клацнемо на елементі «Метка2». Панель Свойства-

Метка1 заміститься панеллю Свойства-Метка2. Змінимо надпис (Caption) «Метка2» на «Термін (роки)». Так само змінимо надписи на інших мітках: «Метка3» на «Річний відсоток», а «Метка4» на «Накопичення (Грн.)».

5. Ще відкриємо панель Свойства–Форма1 і замінимо надпис (Caption) «Форма1» на «Депозит». На цьому закінчуємо створення вигляду форми додатка.

6. Для програмування обираємо подію – перехід фокуса на елемент Текст4, з тим щоб після введення відповідних даних в поля Текст1, Текст2, Текст3 щигликом на елементі Текст4 ініціювати обчислення і вивід результату в поле Текст4.

Клацнемо кнопку **Исследователь программы** на панелі інструментів у вікні проекту. У вікні Программы – Проект1 натиснемо кнопку **Просмотреть код**. Відкриється вікно Проект1–Форма1(Код). Відкриємо список **Общее** і в ньому клацнемо Текст4. Відкриємо інший список **Описание** і клацнемо **GotFocus**.

У вікні Проект1–Форма1(Код) виникнуть рядки

```
Private Sub Текст4_GotFocus()

End Sub,
```

між якими записується код програми.

```
Private Sub Текст4_GotFocus()
Dim B!,T!,p!,S!
    B=Val(Текст1.Text)
    t= Val(Текст2.Text)
    p= Val(Текст3.Text)
    S=B*(1+p/100)^t
    Текст4.Text=S
End Sub
```

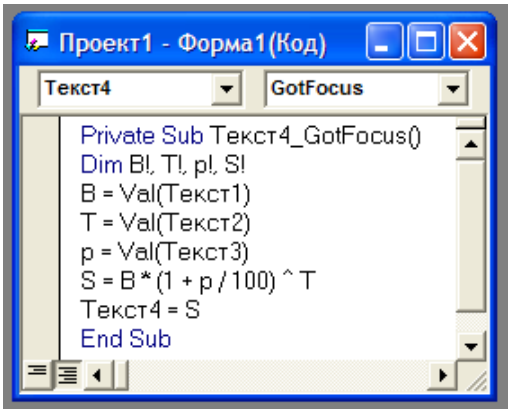



**Рисунок 3.5.** Вікно коду

Можна помітити що коди в тексті і на малюнку дещо різняться. Справа в тому, що оскільки властивість Text є головною для елемента TextBox, то для її використання досить вказати лише ім’я текстового вікна, що й було зроблено.

Результат виконання програми відображено на рис. 3.6.

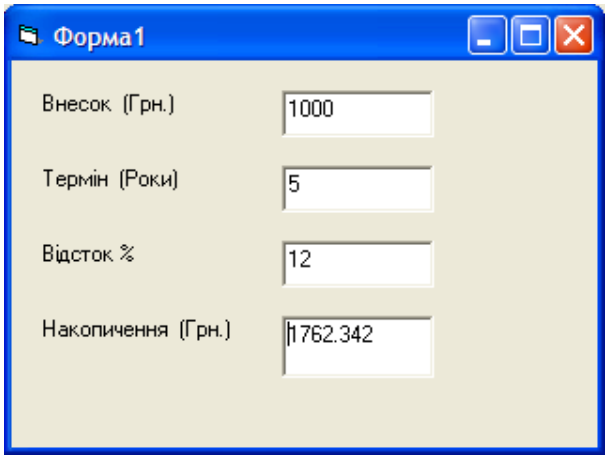



**Рисунок .3.6.** Вікно додатку з результатом

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке змінна? Яким чином можна оголосити змінні?
2. Для чого використовуються ідентифікатори?
3. Чим константа відрізняється від змінної?
4. Які області визначення мають змінні?
5. Як можна визначити до якого типу даних відноситься змінна?
6. Що таке функція? Які з вбудованих функцій ви знаєте?
7. Що таке арифметичні вирази? Який пріоритет встановлено серед операцій?
8. з чого складається логічний вираз? Чим він відрізняється від арифметичного виразу?
9. Які логічні операції існують у Visual Basic?
10. Чи можна оголосити змінні без використання оператора Dim?
11. Яку структуру мають програми на мові Visual Basic?
12. Для чого існують коментарі?
13. Які оператори вводу-виводу ви знаєте? Які вони мають формати? Наведіть приклади.
14. За допомогою яких опереторів у Visual Basic будується таблиця?
15. Які оператори можна використовувати для присвоювання та обміну значеннями? Їх застосування та приклади.

# *Лекція* 4

# ОПЕРАТОРИ УПРАВЛІННЯ

***План лекції:***

*4.1. Оператори розгалуження*

*4.2. Проектування додатка на базі операторів циклу*

*4.3. Оператори умовного циклу*

*4.3. Оператори циклу з лічильником*

## 4.1. ОПЕРАТОРИ РОЗГАЛУЖЕННЯ

В системі програмування Visual Basic оператори в тілі процедури виконуються послідовно один за одним. Для зміни послідовності виконання операторів використовуються оператори управління. До них відносяться умовні оператори та оператори циклу.

**Умовний оператор If…Then…Else**

Умовний оператор використовується для подання розгалуженого обчислювального процесу у коді програми мовою Visual Basic. Умовний оператор If…Then…Else має дві структури: лінійну та блочну.

Синтаксис лінійної структури умовного оператора:

*If <логічний вираз> Then <оператори1> [ Else <оператори2>]*

*Дія*: обчислюється значення логічного виразу, результатом якого може бути одне з двох значень (істина) або (хибність). Логічний вираз складається з операндів логічного типу, між якими містяться знаки логічних операцій і може мати одне з двох значень: True (Істина) або False (Хибність);

оператори 1 виконуються при значенні логічного виразу "істина";

оператори 2 виконуються при значенні логічного виразу "хибність".

**Зауваження! 1**. Оператором 2 знов може бути умовний оператор.
**2.** Вираз у квадратних дужках [ ] може бути відсутній.

Блочна форма оператора If…Then…Else

Синтаксис блочної форми умовного оператора:

*If< логічний вираз 1> Then*
*оператори 1*
*[Else if <логічний вираз2> Then*
*оператори 2]*
*[Else*
*оператори n]*
*End If*

Оператори 1 – це довільна кількість операторів, що виконуються при умові:

Логічний вираз 1 є "істина";

Логічний вираз 2 повертає ненульове значення (істина) або нуль(хибність)

Оператори 2 – це довільна кількість операторів, що виконуються при умові:

Логічний вираз 2 є "істина";

Оператори n – це довільна кількість операторів, що виконуються при інших умовах.

**Приклад.** *Створимо додаток для обчислення платні за працю в нічні години (умову дивіться у Приклад 2 розділу 1).*

Для введення даних і відображення результату використаємо елементи типу TextBox. Для програмування зупинимось на події – втрата фокуса елементом Текст2 (Текст2_ LostFocus()). Програма і вигляд додатку відображені нижче.

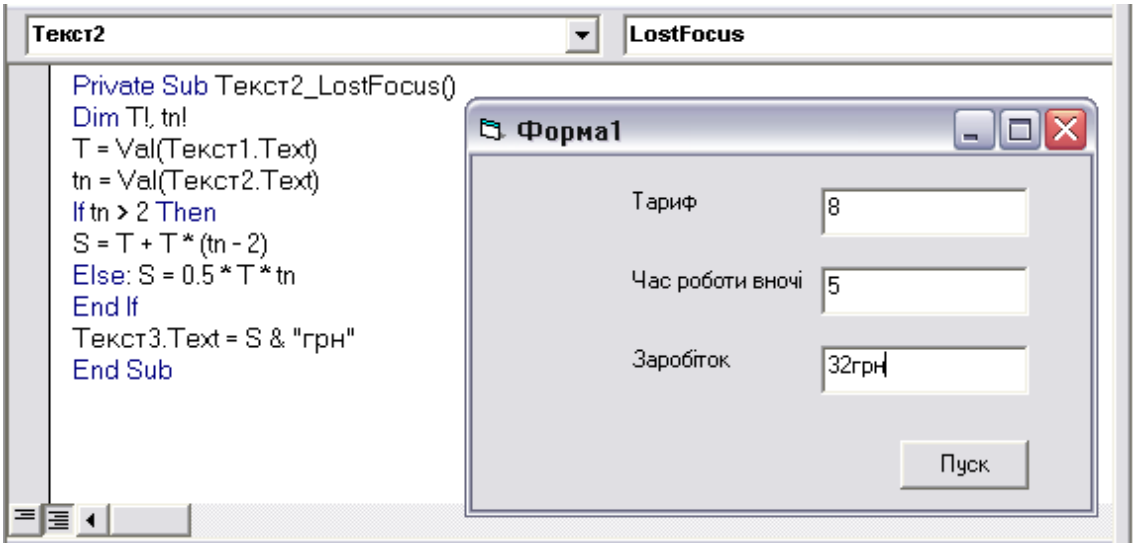



**Рисунок 3.7.** Програма обчислення платні за роботу вночі.

**Оператор Select Case**

**Приклад** використання оператора множинного вибору. *Розглянемо використання оператора множинного вибору Select Case на прикладі обчислення денного заробітку робітника з врахуванням тарифу і тарифного розряду. Код і форма відображені на рис. 3.8.*

На формі розмістимо командну кнопку, яку перейменуємо на «Пуск» та відповідно змінимо напис на ній.

Подією для програмування оберемо клацання лівою кнопки миші на кнопці Пуск.

Для вибору тарифного коефіцієнта за розрядом використаємо оператор Select Case r.

Параметр вибору r може приймати 7 різних значень тож використання множинного оператора вибору значно зручніше ніж багаторазове використання лінійної форми умовного оператора If.

```
If r=1 then k=1
If r=2 then k=1.1
……….
If r=7 then k=2.2

Private Sub Пуск_Click()
Dim Z!, r%, t!, tr!, k!
    T = Val(InputBox("T", "Укажіть тариф"))
    r = Val(InputBox("r", "Тарифний розряд"))
    tr = Val(InputBox("tr", "Відпрацьований час"))
    Select Case r
    Case 1: k = 1: Case 2: k = 1.1: Case 3: k = 1.35:
    Case 4: k = 1.5: Case 5: k = 1.7: Case 6: k = 2:
    Case 7: k = 2.2
    End Select
    Z = t * tr * k
Print "Тариф T=" & T; "  Грн/годину"
Print "Розряд r=" & r
Print "k=" & k
Print " Відпрацьовано tr=" & tr & "  Годин"
Print "Заробіток  Z=" & Z & "  Грн."
End Sub
```

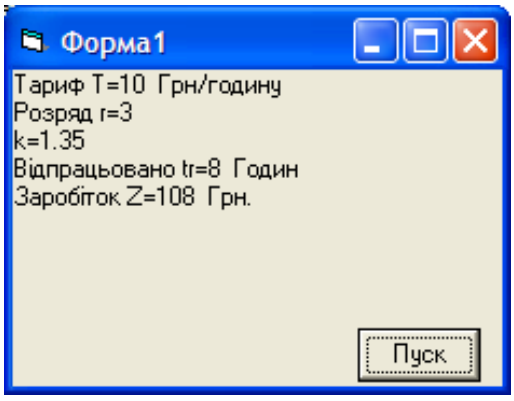



**Рисунок 3.8.** Результат обчислень заробітку

В вище розглянутому додатку запрограмовано подію, якою є щиглик на командній кнопці Пуск, єдиному елементу інтерфейсу встановленому на формі. Введеня даних здійснюється за допомогою функції InputBox, а фіксація їх і результату виконується безпосереднім виведенням на форму за допомогою оператора Print.

## 4.2. ПРОЕКТУВАННЯ ДОДАТКА НА БАЗІ ОПЕРАТОРІВ ЦИКЛУ

Циклічні операціїї, тобто виконання одних і тих же дій багаторазово при різних значеннях величин, які входять до циклу, використовуються в програмуванні доволі часто. Для організації циклу необхідно: а) задати початкове та кінцеве значення параметра циклу – змінної, що змінюється при повтореннях циклу. б) перевірити умову, яка забезпечує вихід з циклу при досягненні мети; в) вказати крок зміни параметру циклу.

Розрізняють цикли, де число повторень заздалегідь відоме (арифметичні) і такі цикли, де число повторень заздалегідь невідоме, але його можна визначити під час виконання циклу (ітераційні).

Якщо число повторень відома заздалегідь, то краще використовувати оператор циклу з лічильником, у протилежному випадку оператор умовного циклу з верхнім або нижнім закінченням. Вміння використовувати циклічні алгоритми є базовими для реалізації додатків з використанням масивів даних і масивів елементів керування. Вони дозволяють краще зрозуміти сутність циклічних поцесів, які зустрічаються під час розвязання різних задач на комп'ютері.

Цикл в програмі можна організувати за допомогою операторів циклу For...Next, While…Wend, Do...Loop.

## 4.3. ОПЕРАТОРИ УМОВНОГО ЦИКЛУ

Умовний цикл одержав свою назву через те, що при кожному повторенні превіряється виконання умови, і рішення про подальші дії приймається залежно від результатів перевірки.

**Оператор циклу While…Wend**

*Синтаксис:*

*While<умова>*

*<Група операторів>*

*Wend*

де умова – логічний вираз, який оцінюється як істина (1), або хибність (0).

*Дія:* Оператор призначений для організації циклу з передумовою, у якому вказана група операторів виконується циклічно до того часу, поки вказана умова не стане хибною. Змінна, що входить в умову, змінюється в тілі циклу. Для дострокового виходу із циклу в разі виконання деякої додаткової умови вживається оператор Exit While.

Допускається використання вкладених циклів. При цьому кількість операторів While і відповідних їм Wend повинна збігатись.

**Приклад.** *Обчислити:*

$$S = \sum_{i=1}^{n} \frac{\cos(i)}{(i+1)^3}; .. n = 10$$

Створимо форму без жодного постійного елемента керування. Подія для програмування – завантаження форми.

```
Private Sub Form_Load()
Dim I%, n%, S!
    N=InputBox("n=")
    I=1: S=0
    While I <=n
        S=S+cos(i)/(I+1)^3
        I=I+1
    Wend
    MsgBox ("S="+Str(S))
    End
End Sub
```

**Оператор циклу Do…Loop**

Поряд з операторами For…Next та While…Wend застосовується більш сучасний оператор циклу Do…Loop, що має два можливих формати: з верхнім і нижнім закінченням.

Першому формату відповідає наступний синтаксис:

*Do {While/Until}<умова>*

*<група операторів >*

*[Exit Do]*

*Loop*

Конструкція {*While/Until*} означає, що використовується тільке одне з ключових слів: або *While* (*Поки*)**,** або *Until* (*До*)**.**

При такій формі запису спочатку здійснюється перевірка умови повторення циклу і якщо вона не виконується, то цикл ні разу не виконується. Фрагменти While/Until мають умови, що визначають вихід з циклу чи його повторення. В конструкції While записується умова, хибність якої визначає вихід з циклу – перехід до оператора, що йде за оператором циклу, тобто після слова Loop. Хибність логічного виразу (умови) в конструкції Until визначає повторення циклу, яке здійснюється до її виконання, після чого відбувається вихід з циклу.

Другому формату відповідає синтаксис

*Do*

*<група операторів>*

*[Exit Do]*

*Loop {While/ Until} <умова>*

де <умова> – логічний вираз, який може приймати два значення: істина (не 0), або хибність (0);

While – виконання циклу, поки умова істина;

Until – виконання циклу, поки умова хибна:

*Дія:* У цій формі запису спочатку виконується група операторів (тіло циклу), після чого виконується перевірка умови повторення циклу. Таким чином, цей цикл завжди виконується  хоча б один раз.

Цикл з нижнім закінченням вживається тоді, коли оператори тіла циклу повинні виконуватися хоча б один раз, а з верхнім – навіть жодного.

Серед операторів тіла циклу може бути оператор **Exit Do**, що забезпечує негайний вихід з циклу (до досягнення ключового слова **Loop**)

**Приклад.** *Обчислити суму спадного ряду, використовуючи усі можливі варіанти використання оператора Do…Loop.*

$$S = \sum_{k=1}^{\infty} \frac{\sin(k)}{k^3 + 3}$$ з точністю ε=10-5

Наведемо 4 варіанти реалізації циклу Do Loop для обчислення суми ряду.

Варіант 1

```
Private Sub Комманда1_Click()
Dim s!, e!, h! , k%
    s=0: k=1: e=0.00001
    Do
        h=sin(k)/(k^3+3)
        s=s+h
        k=k+1
    Loop While Abs(h) >e
Print “k=”; k, “s=”; s, “h=”, h
End Sub
```

Варіант 2

```
Private Sub Комманда1_Click()
Dim s!, e!, h! , k%
    s=0: k=1: e=0.00001:h=1
    Do While Abs(h) > e
        h=sin(k)/(k^3+3)
        s=s+h
        k=k+1
    Loop
Print “k=”; k, “s=”; s, “h=”, h
End Sub
```

Варіант 3

```
Private Sub Комманда1_Click()
Dim s!, e!, h! , k%
    s=0: k=1: e=0.00001
    Do
        h=sin(k)/(k^3+3)
        s=s+h
        k=k+1
    Loop Until Abs(h) >e
Print “k=”; k, “s=”; s, “h=”, h
End Sub
```

Варіант 4

```
Private Sub Комманда1_Click()
Dim s!, e!, h! , k%
    S=0: k=1: e=0.00001:h=1
    Do Until  Abs(h) < e
        h=sin(k)/(k^3+3)
        s=s+h
        k=k+1
    Loop
Print “k=”; k, “s=”; s, “h=”, h
End Sub
```

**Приклад**. Розглянемо задачу економічного характеру.

*При відвідуванні магазину покупець робить покупки в процесі знайомства з товаром, тобто заздалегідь не відомо, які товари будуть куплені. Потрібно визначити вартість усіх покупок. Передбачити випадок, коли немає жодної покупки.*

Для визначення загальної вартості покупок необхідно врахувати вартість кожного виду товару відібраного покупцем, для чого потріно знати його ціну і кількість. Продавці звичайно ведуть облік проданих товарів, тому крім ціни і кількості передбачимо введення назви товару (або його коду). Кодом може слугувати порядковий номер товару в спискові товарів цього магазину чи відділу. Отримані відомості бажано зберегти на дискові в файлі послідовного чи безпосереднього доступу. Маючи це на увазі, розмістимо на формі три елементи ListBox (Списки), в які будемо заносити дані про покупки.

В Список1 запишемо коди, в Список2 – Ціни (Грн.), в Список3 –кількість. Над списками розмістимо відповідні мітки. Вводити дані будемо за допомогою функції InputBox, запитання до покупця і підсумок будемо виводити за допомогою функції MsgBox. Розмістимо на формі ще командну кнопку, яку перейменуємо на Обчислити, змінивши відповідно надпис на ній. За подію для програмуван-

ня виберемо щиглик на кнопці Обчислити. Оскільки Ціна і Вартість S повинні подаватися в горшових одиницях, то оголошуємо їх As Currency, кількість оголошуємо числом одинарної точності (As Single), код товару відображується цілим числом (As Integer).

Код процедури має вигляд:

```
Private Sub Обчислити_Click()
Dim Код As Integer, Ціна As Currency, S As Single, i As Integer
    S=0: ‘ Загальна вартість
    i=0 ‘ номер покупки
    Do While MsgBox(“Будуть ще покупки?”, vbQuestion +  _ vbYesNo, “Покупки”)=vbYes
        i=i+1
        Код=Val(InputBox(“Введіть ціну”, i & “ –го товару”))
        Ціна=CCur(InputBox(“Введіть ціну”, i & “ –го товару”))
        Кількість=Val(InputBox(“Введіть ціну”, i & “ –го товару”))
        Список1. AddItem Код
        Список2. AddItem Ціна
        Список3. AddItem Кількість
        S=S+Ціна*Кількість
    Loop
MsgBox “Вартість всіх покупок “  & Format( S, “0.00”) & “Грн.”
End Sub
```

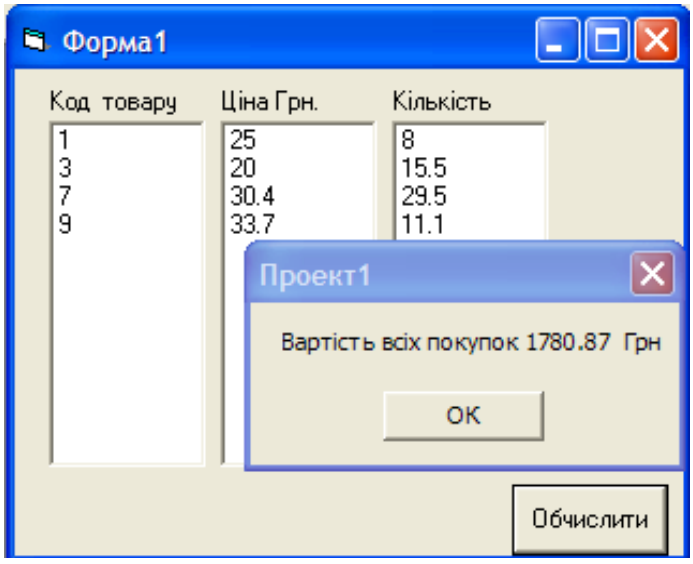



**Рисунок 3.10.** Результат дії додатку Покупки

## 4.4. ОПЕРАТОР ЦИКЛУ FOR...NEXT

*Синтаксис:*

*For x=x0 To xn [ Step h ]*

*<Група операторів>*

*[Exit For]*

*Next x,*

де x – параметр циклу;

$x_o$, $x_n$, h – арифметичні вирази, значення яких визначають:

$x_0$ – початкове значення параметра циклу;

$x_n$ – кінцеве значення параметра циклу;

h – крок, з яким змінюється параметр циклу. Якщо h=1,то Step h не пишеться. Exit For використовується в разіі необхідності дострокового виходу з циклу.

*Дія оператора*. При виконанні оператора циклу For...Next в першу чергу обчислюється (якщо це необхідно) і запам'ятовується початкове значення ($x_0$), кінцеве значення ($x_n$) та крок зміни (h) параметра циклу. Параметру циклу присвоюється початкове значення ($x=x_0$) і перевіряється умова, чи перевищує значення параметра циклу x значення xn. Якщо значення параметру циклу знаходиться в інтервалі між початковим та кінцевим значенням, виконуються оператори, що знаходяться між операторами For та Next (тіло циклу). Далі оператор циклу збільшується на величину h. Дії повторюються до того часу, поки параметр циклу x не стане більшим за $x_n$. В такому випадку управління передається на виконання оператора, що стоїть в програмі після оператора Next x.

При виході з циклу достроково (наявність оператора Exit For) значення параметра циклу дорівнює останньому значенню в середині циклу.

При виході з циклу через оператор Next значення параметру дорівнює останньому його значенню плюс величина кроку зміни параметру.

Дозволяється використання циклу в циклі ( вкладених циклів ). В цьому випадку внутрішній цикл повинен повністю знаходитися в тілі зовнішнього циклу.

**Приклад.** *Скласти схеми та програми для варіантів (а) та (б) обчислення значень a, p та y. Результати отримати у вигляді таблиці значень.*

$$p = \sqrt{x^2 + |\cos y|} + e^x; \quad a = \sqrt{17.89} \cdot x + \frac{c + \sin d}{c - \cos d}$$

$$y = \begin{cases} arctg(a + x), a > x \\ \ln|a^2 + x|, a <= x \end{cases} \qquad d = 4{,}7,\ c = 1{,}854$$

а) x={1; 1,5; 2, ... 5}, b) x={1,5; 4,9; 8,36; -0,87 }

Параметри початкового ($x_0$=1.5), кінцевого ($x_n$=5) значень та кроку (h==0.5) визначаються із вказанної послідовності значень змінної x (варіант (a)).

Для введення змінних d, c, $x_0$, $x_n$, h створюємо масив z (використовуємо функцію Array)

Програма до варианту a).

```
Private Sub Комманда1_Click()
Rem Програма до варіанту a)
Dim a!,y!,p!
        Z=Array(4.7,1.854,1,5,5,0.5)
        D=z(0):c=z(1): x0=z(2): xn=z(3); h=z(4)
    Print Tab (10);"Таблица"
    Print Tab(10); String$(14,"*")
    Print Tab(3);"x";Tab (12);"a";_
        Tab(24);"y";Tab (48);"p"
    Print String$(52,"*")
    For x=x0 To xn Step 0.5
        A=sqr(17.89)*x+(c+sin(d))/(c-cos(d))
        If a>x then
        Y=Atn(a+x)
    Else
        Y=log(ABS(cjs(y)))+exp(x)
        Print x;Tab(8);a; Tab(19);y; Tab(33); p
    Next x
Print string$(52,"*")
End sub
```

Програма до варианту b)

Розглядаємо x як масив чисел, тобто x={Xi}, i=1…n (n=4)

```
Option Base 1
Private  Sub Комманда1_Click()
Rem програма по варіанту b)
Dim a!, y!, p!
        Z=Array(4.7,1.854)
        D=z(1):c=z(2)
        X=Array(1.5,4.9,8.36, -0.87)
        n=InputBox("n=")
    Print Tab(10);"Таблиця"
    Print Tab(10);String$(14,"*")
    Print Tab(3); "x"; Tab(12); "a";_
        Tab(24); "y";  Tab(40); "p"
    Print String$(53,"*")
```

```
    For I=1 To n
        A=Sqr(17.89)*x(i)+(c+sin(d))/(c-cos(d))
        If a>x(i) then
        Y=Atn(a+x(i))
    Else
        Y=log(abs(a+x(i)))
    End If
        P=Sqr(abs(cos(y)))+exp(x(i))
        Print x(i); Tab(19);y; Tab (33);p
    Next I
Print String$(53,"*")
End Sub
```

**Приклад**. *Скласти програму для обчислення Z!*

Реалізуємо алгоритм блок-схема якого відображена на рис.1.7.

Розмістимо на формі лише один елемент управління – командну кнопку, ім'я якої і напис на якій змінимо з CommandButton1 на Пуск.

Запрограмуємо подію – щиглик (Click) на кнопці Пуск. Для введення Z використаємо оператор InputBox(), для виведення результату – оператор MsgBox().

```
Private Sub Пуск_Click()
Dim Z %, P%, i%
    Z=Val(Input("Z=?","Задайте  Z")
    P=1
    For i=1 to Z
        P=P*i
    Next i
MsgBox="Z!=" & P
End Sub
```

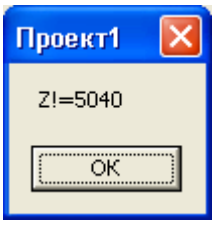



**Рисунок 3.9.** Результат обчислення Z! для Z=7.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. У яких випадках використовують розгалужені алгоритми? Наведіть приклади.
2. Яке призначення має умовний оператор?
3. Порівняйте за призначенням лінійну та блочну форми умовного оператора?
4. З якою метою можна застосовувати функцію MsgBox?
5. Придумайте три задачі з повсякденного життя, де було б використано умовний оператор.
6. Яке призначення мають оператори циклу?
7. У яких випадках використовуються оператори умовного циклу? Наведіть приклади.
8. За допомогою яких операторів можна організувати цикл?
9. Які типи циклів розрізняють?
10. Який синтаксис має оператор циклу While…Wend?
11. Який синтаксис має оператор циклу Do…Loop?

# *Лекція* 5

# МАСИВИ

***План лекції:***

*5.1. Статичні масиви*

*5.2. Динамічні масиви*

*5.3. Функція створення масиву Array*

*5.4. Використання одновимірних маси*

*5.5. Використання двовимірних масивів*

*Масив* – це впорядкований набір даних одного типу, кожен з яких має свій порядковий номер, що називається індексом. Розрізняють статичні та динамічні масиви.

Кожен елемент масиву має свій номер (індекс). Причому таких номерів може бути декілька. Кількість різних індексів визначає вимірніть масиву. Бувають одновимірні (вектори), двовимірні (матриці), тривимірні і т.д. кожен індекс може набувати різних значень з визначеного діапазону. Мінімальне $N_{min}$ і максимальне $N_{max}$ значення діапазону зміни індексу називається *граничною парою*. Кількісь різних знвчень індексу визначає кількість елементів $K$ в масиві. Так, для одновимірного масиву

$$K=N_{max}-N_{min}+1$$

При використанні масивів в Visual Basic нумерація масивів за замовчуванням починається з нуля. В математиці елементи масивів нумеруються з одиниці (1). Оператор Option Base дозволяє задати індексацію з 1.

*Синтаксис: Option Base 1*

---

**Зауваження!** Цей оператор повинен знаходитись в секції (General) (Declrations) контейнеру (форми, модулю, класу). Допустимими значеннями для Option Base є тільки 0 або 1.

---

## 5.1. СТАТИЧНІ МАСИВИ

Для об’явлення масивів використовують оператор Dim.

*Синтаксис:*

*Dim <ім'я> (n) As <тип>,*

де <ім'я> – довільний ідентифікатор;

n – натуральне число, яке вказує на розмірність масиву (max кількість елементів в масиві);

**Приклад.** Dim a(5) As Single – означає, що маємо 5 значень в масиві a (a0,a1,a2,a3,a4) одинарної точності з плаваючою десятковою комою.

Для встановлення інших границь масиву необхідно використати наступний синтаксис:

*[Static | Public| Dim|<Імя >(<нижня границя > To <верхня границя>)*

Visual Basic дозволяє також створювати багатовимірні масиви. При оголошенні багатовимірного масиву, границі кожного виміру розділяються комами.

*Синтаксис:*

*Dim<Ім'я> (<нижня границя> To <верхня границя>, ..., <нижня границя> To <верхня границя>)*

**Приклад.** Dim X(10 To 80, 1 To 25, 1956 To 2050) – задано тривимірний масив X, де 10,1,1956 – нижні границі; 80, 25, 2050 – верхні границі.

## 5.2. ДИНАМІЧНІ МАСИВИ

Інколи при оголошенні масиву його розмір невідомий. В такому випадку слід оголошувати динамічний масив, що дозволяє його розмір, або розмірність змінювати під час виконання програми.

Динамічний масив створюється в два етапи. Спочатку масив визначають в секції (General) (Declarations) контейнеру (форми, модуля) без указання розміру.

**Приклад**. '(General) (Declarations)

*Dim A( ) As Variant*

Потім за допомогою оператора ReDim встановлюють фактичний розмір масиву.

*Синтаксис: ReDim [ Preserve ] <ім'я> (границі) [As тип даних]*

Оператор ReDim використовується тільки в процедурах. Тип даних вказувати необов'язково, особливо, коли тип вже визначений оператором Dim.

**Приклад**. '(General) (Declarations)

```
Dim A( ) As Variant
Private Sub Command 1_Click()
    ReDim A(50To10)
    Тіло процедури
End Sub
```

Не допускається оголошувати масив з даними одного типу, а потім використати ReDim для приведення до іншого типу, за виключенням випадку, коли масив містить змінні типу Variant.

При застосуванні оператора ReDim виникає небезпека втрати його змісту, оскільки після зміни розмірності елементам масиву присвоюються значення за

замовчуванням. Щоб не було втрати змісту необхідно застосовувати ReDim разом з ключовим словом Preserve.

Приклад. '(General) (Declarations)

```
Dim A( ) As Variant
Private Sub Command 1_Click()
    ReDim Preserve A(50 To 15)
    Тіло процедури
End Sub
```

**Зауваження!** При використанні ключового слова Preserve можна змінювати тільки верхню границю.

**Приклад**. ‘(General) (Declarations)

```
Dim A( ) As Variant
Private Sub Command 1_Click
    ReDim A (10 To 20)
    ReDim Preserve A(10 To 25)’ Вірно
    ReDim Preserve A(15 To 25)’ Помилка
    Тіло процедури
End Sub
```

## 5.3. ФУНКЦІЯ СТВОРЕННЯ МАСИВУ ARRAY

*Призначення.*

Функція створює новий масив типу Variant, який буде відразу заповнений зазначеними елементами. Якщо Ви хочете створити масив іншого типу, то оголошуйте його за допомогою оператора Dim.

*Синтаксис: Array(<список елементів масиву>)*

Список елементів масиву типу Variant, в якому елементи розділяються комами. При відсутності списку, створюється масив нульової довжини.

**Зауваження!** Зверніть увагу, що нумерація елементів масиву починається з 0. Якщо хочете почати з 1, використовуйте Option Base 1.

**Приклад.** *Створити масив А з елементами 5, 7, 12. Змінній В присвоїти значення елемента масиву А, який стоїть на третьому місці.*

```
Dim A,B As Integer ’Об’явлені цілочисельні змінні,
A=Array(5,7,12)   ’Створено масив з трьох елементів. Індексація елементів масиву починається з 0.
B=A(2)            ’В передається другий елемент масиву А.
Print B           ’Повертається 12.
```

## 5.4. ВИКОРИСТАННЯ ОДНОВИМІРНИХ МАСИВІВ

Описування одновимірного масиву здійснюється оператором Dim такими способами:

*Dim Ім'я_змінної ($N_{max}$)[<As Тип_елементів>]*

*Dim Ім'я_змінної ($N_{min}$ To $N_{max}$)[<As Тип_елементів >]*

У першому випадку вважається, що $N_{min}$=0. якщо потрібно, щоб нижня межа діапазону зміни індексу у всіх масивів, описаних першим способом, була не 0, а 1, то перед першою процедурою коду потрібно помістити оператор Option Base 1.

У цьому разі верхня межа буде визначати кількість елементів у масиві.

Доступ до елемента масиву здійснюється за допомогою імені масиву й індексу. Як індекс може вживатися вираз.

Слід мати на увазі, що будь-яка обробка масивів здійснюється поелементно.

Наведемо приклади типових алгоритмів обробки одновимірних масивів.

**Приклад**. *Введення одновимірного масиву B розміром N*:

```
Dim B(N)
For i=1 To N
    B(i)=Inputbox("Введіть B("& i &")" & "елемент масиву", "Вікно вводу вектора")
Next i
```

**Приклад**. *Виведення одновимірного масиву B(N)*:

```
For i=1 To N
    Print B(i)
Next i
```

**Приклад**. *Підсумовування елементів масиву B(N)*:

```
S=0
For i=1 To N
    S=S+B(i)
Next i
```

**Приклад**. *Підсумовування двох масивів A і B розміром N*:

```
For i=1 To N
    C(i)=A(i)+B(i)
Next i
```

**Приклад**. *Визначити кількість елементів масиву B(N), задовольняючих задані умови, наприклад, B(i)>T, де T – задане число*:

```
K=0
For i=1 To N
    If B(i)<=T goto м1
        K=K+1
    м1
Next i
```

**Приклад**. *Підсумовування елементів масиву B(N), задовольняючих заданій умові (B(i)>T)*:

```
S=0
For i=1 To N
    If B(i)>=T Then S=S+B(i)
Next i
```

**Приклад**. *Формування масиву B(M) із елементів іншого масиву A(N), задовольняючих заданій умові (A(i)>T)*:

```
j=0
For i=1 To N
    If A(i)>T Then
        j=j+1
        B(j)=A(i)
    End If
Next i
```

**Приклад**. *Пошук максимального елементу в масиві A(N) із запам'ятовуванням його положення (індексу) в масиві*:

```
P=A(1)
K=1
For i=2 To N
    If P<=A(i) Then
        P=A(i):K=i
    End If
Next i
```

**Приклад**. *Об'єднання двох масивів A і B розміром N в один масив C розміром 2N з чергуванням елементів початкових масивів*:

```
For i=1 To N
    C(2*i-1)=A(i)
    C(2*i)=B(i)
Next i
```

**Приклад**. *Упорядкування масиву A(N) в порядку зростання елементів*:

```
For i=1 To N-1
    P=A(i):K=i
    For j=i+1 To N
        If A(j)<=P Then
            P=A(j):K=j
        End If
    Next j
    A(k)=A(i):A(i)=P
Next i
```

## 5.5. ВИКОРИСТАННЯ ДВОВИМІРНИХ МАСИВІВ

Часто доводиться обробляти великі груп даних.

Перед створенням двовимірний масив також оголошують, при цьому залежно від місця оголошення масив може бути як локальним, так і глобальним. Якщо кількість елементів масиву відомо, то це масив з фіксованими межами. У випадку змінної кількості елементів створюють динамічний масив. Для оголошення масиву використовують оператор

*Dim <ім'я масиву> (<число1>,<число2>)[<As тип даних>]*

де імя масиву – імя змінної типу «масив», число1 – розмір масиву (число рядків), число2 – розмір масиву (число стовпців), тип даних – тип даних масиву, за замовчуванням – тип Variant.

Кожен елемент масиву має свій індекс (місце розташування в масиві), що складається з номера рядка та номера стовпця. По замовчуванню значення нижньої ежі масиву дорівнюється нулю.

Наприклад, елементи двовимірного масиву A(2,3), що складається із трьох рядків і чотирьох стовпців, характеризуються такими індексами:

A(0,0); A(0,1); A(0,2); A(0,3)

A(1,0); A(1,1); A(1,2); A(1,3)

A(2,0); A(2,1); A(2,2); A(2,3)

Але, якщо в розділі General модуля програми помістити оператор Option Base 1, то елементи масиву будуть нумеруватись з одиниці. При необхідності базовий індекс при оголошенні масиву можна змінювати шляхом використання ключового слова To: *Dim StrMyArray(1 To 3, 1 To 4) As String*.

Під час роботи з масивами використовують оператори циклу For…Next, при цьому лічильник повторень циклів пов'язується з індексами елементів.

В якості індексів можна використовувати не тільки константи, але і змінні, що дає можливість задавати дії над будь-якими елементами масиву. В пам'яті елементи двовимірного масиву розташовуються по рядках.

В мові VB 6.0 не визначені операції із матрицями, тому будь-яка обробка матриць здійснюється поелементно.

Наведемо приклади типових алгоритмів обробки матриць.

**Приклад**. *Введення двовимірного масиву B розміром 4*x*5 можна здійснити за допомогою опереторів*:

```
Dim B(4,5)
    For i=1 To 4
        For j=1 To 5
```

```
B(i,j)=Inputbox("Введіть B("& i &","& j &")" & "елемент масиву", "Вікно вводу матриці")
        Next j
Next i
```

Введеня двовимірного масиву в наведеній програмі здійснюється по рядкам.

**Приклад.** *Виведення двовимірного масиву по рядкам*

```
For i=1 To 5
    For j=1 To 5
        Print B(i,j)
    Next j
    Print
Next i
```

В даній програмі виведення нового рядка масиву здійснюється в новий рядок форми.

**Приклад.** *Сумування елементів масиву. Для двовимірного масиву B розміром NxM необхідно обчислити*

$$S = \sum_{i=1}^{N} \sum_{j=1}^{M} b_{ij}$$

```
S=0
For i=1 To N
    For j=1 To M
        S=S+B(i,j)
    Next j
Next i
```

**Приклад.** *Підсумовування матриць. Для двовимірних масивів A та B розміром N*x*M. Необхідно обчислити*

$$C_{ij} = a_{ij} + b_{ij},\ i=1,2,...,N\ j=1,2,...,M$$

```
For i=1 To N
    For j=1 To M
        C(i,j)=A(i,j)+B(i,j)
    Next j
Next i
```

**Приклад.** *Підсумовування діагональних елементів матриці. Обчислення ліду матриці. Для матриці B розміром N*x*N*

$$S_i = \sum_{i=1}^{N} b_{ii}$$

```
REM Обчислення сліду матриці
S=0
For i=1 To N
    S=S+B(i,j)
Next i
```

**Приклад**. *Підсумовування елементів рядків матриці. Необхідно обчислити суму елементів кожного рядка матриці B розміром NxM. Результат отримати у вигляді вектора D, тобто обчислити*

$$D_i = \sum_{j=1}^{N} b_{ij}$$

```
REM Сумування матриці по рядкам
For i=1 To N
S=0
    For j=1 To M
        S=S+B(i,j)
    Next j
D(i)=S
Next i
```

**Приклад**. *Транспонювання матриці. Необхідно замінити рядки матриці її стовпцями, а стовпчики – рядками, тобто обчислити*

$$a_{ji} = b_{ij},\ i=1,2,...,N\ j=1,2,...,M$$

```
REM Транспонування матриці
For i=1 To N
    For j=1 To M
        B(i,j)=A(j,i)
    Next j
Next i
```

**Приклад**. *Множення матриці на вектор. Для обчислення добутку C матриці A розміром NxM на вектор B розміром M, тобто обчислити*

$$C_i = \sum_{j=1}^{N} a_{ij} \cdot b_j$$

```
REM Множення матриці на вектор
For i=1 To N
    For j=1 To M
        S=S+A(i,j)*B(j)
    Next j
    C(i)=S
Next i
```

**Приклад**. *Множення матриці на матрицю. Для множення матриці A(N,K) матрицю B(K,M) необхідно обчислити*

$$C_{ij} = \sum_{l=1}^{K} a_{il} \cdot b_l,\quad i=1,2,...,N\ j=1,2,...,M$$

```
REM Множення матриці на матрицю
For i=1 To N
    For j=1 To M
```

```
    S=0
        For l=1 To K
            S=S+A(i,l)*B(l,j)
        Next l
    C(i,j)=S
    Next j
Next i
```

**Приклад.** *Перестановка рядків матриці:*

*а) з використанням допоміжної змінної P*

```
REM Перестановка рядків
For K=1 To M
    P=A(i,K)
    A(i,K)=A(j,K)
    A(j,K)=P
Next K
```

*б) з використанням допоміжного масиву C*

```
REM Перестановка рядків
For K=1 To M
    C(K)=A(i,K)
Next K
For K=1 To M
    A(i,K)=A(j,K)
Next K
For K=1 To M
A(j,K)=C(K)
Next K
```

**Приклад.** *Пошук мінімального елемента матриці:*

```
P=A(1,1)
K=1 : L=1
For i=1 To N
    For j=1 To M
        If P<=A(i,j) Then Goto metka1
        P=A(i,j)
        K=i
        L=j
        metka1:
    Next j
Next i
```

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке масив? З якою метою використовуються масиви даних?
2. Які існують види масивів? Наведіть пиклади.
3. Що означає поняття вимірність масиву?
4. Наведіть приклади одновимірних, двовимірних і тривимірних масивів.
5. Як здійснюється введення та виведення елементів матриці на мові Visual Basic?
6. Введення-виведення елементів одновимірного масиву. Наведіть приклади.
7. Наведіть приклади формування елементів одновимірного масиву на довільному проміжку за допомогою функції RND.
8. Яким чином формується масив по умові. Наведіть приклади.
9. Яким чином можна обробити масив, відсортувати його елементи?
10. Наведіть приклади формування елементів двовимірного масиву на довільному проміжку за допомогою функції RND.
11. Який алгоритм визначення найбільшого та найменшого елементів матриці?
12. Для чого використовується функція Option Base?
13. Яку пару елементів масиву можна назвати гранічною?
14. Для чого використовується ключове слово Preserve?
15. За дапомогою якого оператору можна оголосити масив типу Variant?

# *Лекція* 6

# МОДУЛЬНЕ ПРОГРАМУВАННЯ

***План лекції:***

*6.1. Створення процедур (підпрограм) загального призначення*

*6.2. Процедури типу Function*

*6.3. Процедури типу Sub*

## 6.1. СТВОРЕННЯ ПРОЦЕДУР (ПІДПРОГРАМ) ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ

В стандартних модулях існують процедури загального призначення, які можуть бути викликані з будь-якого місця в програмі. Процедури загального призначення близькі до внутрішніх операторів та функцій. В Visual Basic вони викликаються за іменем, можуть мати аргументи і кожна з них виконує конкретні дії. Процедури загального призначення дозволяють економити час, роблять програми більш компактними, зручними в перегляді (прочитанні), спрощують роботу програм.

В стандартному модулі можна створити три види процедур загального призначення:

1. Процедури типу **Function** (функція). Такі процедури викликаються з інших процедур. Вони можуть мати аргументи, через які одержують вхідні значення також повертати значення у вигляді імені. Зазвичай використовуються для обчислень.

2. Процедури типу **Sub** (підпрограми). Вони також викликаються за ім'ям з інших процедур, можуть мати аргументи, одержувати вхідні значення, виконувати відповідні дії і повертати значення. На відміну від функцій процедури Sub використовуються для одержання або обробки вхідних даних і відображення вихідних даних або встановлення властивостей значень.

3. Процедури типу **Property** (Властивість). Такі процедури використовуються для створення властивостей, які визначаються користувачем у програмах, і маніпулювання ними. Це корисно, коли вимагається створити засіб, який дозволяє використовувати засоби управління Visual Basic.

Щоб набрати будь-яку процедуру загального призначення необхідно:

У меню **Разработать** вибрати команду **Добавить модуль** (**Add Module**).

Відкриється вікно Add Module. Вибрати **Новый** та натиснути кнопку **Open** (**Открыть**). У вікні Code з'явиться новий стандартний модуль.

Ввести в стандартному модулі текст процедури.

## 6.2. ПРОЦЕДУРИ ТИПУ FUNCTION

*Синтаксис:*

*Function ім'я (x1 [As тип], x2 [As тип],... xn [As тип]) As Тип*

*блок операторів*

*ім'я = A*

*[оператори ]*

*[Exit Function ]*

*End Function*

де ім’я – ідентифікатор функції;

x1, x2, x3..., xn - формальні параметри, які можуть бути:

1) змінними;

2) масивом;

Exit Function – достроковий вихід із процедури.

Звернення до функції здійснюється за ім’ям:

*ім'я(a1,a2, ...,an),*

де a1, a2,..., an – фактичні аргументи.

Фактичними аргументами можуть бути:

- змінні (прості та з індексами);
- масив;
- константа;
- арифметичний вираз.

**Зауваження**! Щоб передати у процедуру масив, достатньо після імені вказати порожні дужки X().

*Дія:*

При зверненні до процедури:

1) формальні параметри визначаються фактичними аргументами;

2) виконується процедура, в якій ім’я приймає значення;

3) значення передається в головний модуль.

**Приклад.** *Обчислити P=arctg a+4.7b, де a – сума компонентів вектора x; b – сума компонентів вектора y; x(1.5; -6.7; 8.35; -1.9; 0.7)*

*Компоненти вектора y обчислюються за формулою*

$$Yi=xi/\cos i$$

Нехай ім’я процедури Function SV.

1. Відкрити новий проект.

2. За допомогою елемента CommandButton створити командну кнопку в формі.

3. Змінити стандартний напис на командній кнопці (властивість Caption) на Пуск.

4. Обов'язково встановити для властивості форми Autoredraw значення Да (True).

5. Двічі клацнути по кнопці Пуск у формі. У вікні Форма1(Код) з'явиться заголовок процедури та його кінець:

```
Private Sub Команда1_Click()

End Sub
```

6.Після заголовку процедури набрати текст(тіло програми).

```
Option Base 1
Dim y!
Private Sub Команда1_Click()
Dim a!,  b!,  p!
    N=InputBox("n=")
    ReDim  y(n)
    X=Array(1.5, -6.7, 8.35, -1.9, 0.7)
    Print "Вектор y"
    For i=1 to n
        Y(i)=x(i)/cos(i)
        Print y(i);Spc(3);
    Next i: Print
    A=SV(5,x)
    B=SV(n,y)
    P=Atn(a)+4.7*b
Print Tab(5);"Результати обчислень"
Print Spc(5);"a="; a; Spc(5);"b=" ; b; Spc(5); "p="; p
End sub
```

7. У меню Разработать обрати команду **Добавить модуль**. З'явиться вікно Add Module. Вибрати **Новый** і натиснути кнопку **Открыть.**

8. З'явиться вікно Проект1-Модуль 1(Код).

Надрукуємо ім'я процедури Function SV(n,z) і натиснемо **Enter**. З'явиться *End Function* (кінець процедури).

9. Після заголовку процедури набрати текст(тіло процедури).

```
Function SV(n,z)
S=0
For i=1 to n
S=s+z(i)
Next i
SV=s
End Function
```

10. Запустити команду на виконання, натиснувши кнопку **Начать** на панелі інструментів **Стандартная**.

11. У вікні Форма1 клацнути по командній кнопці **Пуск**. Ввести розмір масиву *n*. (*n*<=5=розміру массиву *x*).

Результат виконання процедури буде показаний у вікні форми (рис. 6.1).

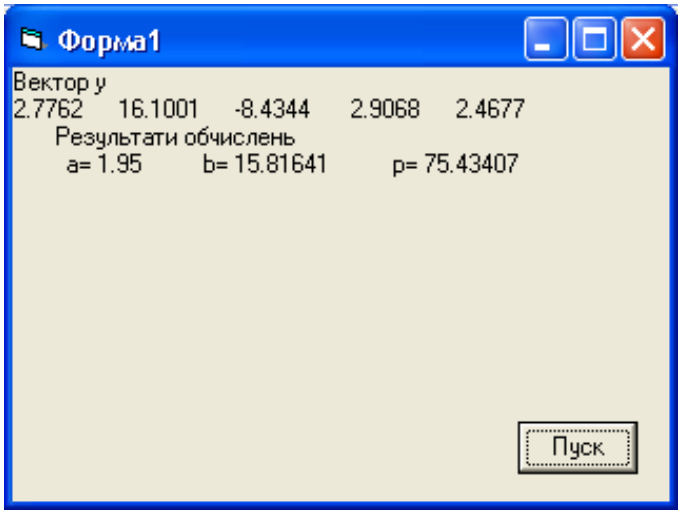



**Рисунок 6.1.** Вікно форми з результатом виконання прикладу.

12. Для закінчення розрахунків необхідно клацнути по кнопці **Конец** на панелі інструментів.

13. Зберегти проект, клацнувши **Сохранить** (**Save Project**).

## 6.3. ПРОЦЕДУРИ ТИПУ SUB

*Синтаксис:*

*Sub ім'я (x1 [As mun], x2 [As mun],... xn [As mun])*

*блок операторів*

*[Exit Sub ]*

*End Sub*

де ім’я – довільний ідентифікатор процедури;

x1, x2, x3,..., xn – формальні параметри, які умовно поділяються на 2 групи:

- параметри, що визначаються фактичними аргументами (при звернені до процедури)
- параметри, що визначають фактичні аргументи (після виконання процедури)

і ті і інщі можуть бути:

1) змінною;

2) масивом;

3) константою;

4) арифметичним виразом.

Exit Sub – достроковий вихід з процедури.

Звернення до процедури – функції здійснюється за допомогою оператора Call

*Синтаксис:*

*Call ім'я(a1,a2, ...,an),*

де a1, a2,...,an – фактичні аргументи.

**Зауваження!** Щоб передати в процедуру масив, достатньо після імені його вказати пусті дужки X().

**Приклад.** *Задані матриці $A_{nm}$ та $B_{mn}$ (n>m)/*

*Нехай елементи матриць знаходяться на проміжку [f1,f2], де f1=-20; f2=20. Обчислити матрицю C=A*B. Формування матриці та знаходження суми елементів матриці оформити у вигляді процедури (підпрограми).*

1. Відкрити новий проект.

2. Створити командну кнопку в формі.

3. Дати нову назву цій кнопці Пуск.

4. Обов’язково встановити для властивості форми Autoredraw значення Да(True).

5. Двічі клацнути по кнопці Пуск у формі. У вікні Форма1(Код) (Code) з’явиться заголовок процедури та його кінець:

*Private Sub Комманда1_Click()*

*End Sub*

Після заголовку процедури набрати текст (тіло програми)

```
Option Base 1
Dim a!(), b!(), c!(), S!
Private Sub Комманда1_Click()
    Dim n%, m%, i%, j%, k%, f1!, f2!, p!
    m = InputBox("m=")
    n = InputBox("n=")
    l = InputBox("L=")
    f1 = Val(InputBox("f1="))
    f2 = Val(InputBox("f2="))
    ReDim a(m, n), b(n, l), C(m, l)
Print Tab(10); "Матриця A"
    Call Form(m, n, f1, f2, a!())
```

```
Print Tab(10); "Матриця B"
    Call Form(n, l, f1, f2, b!())
Print Tab(10); "Матриця C=A*B"
    For i = 1 To m
        For j = 1 To l
            Call Sum(i, j, n, S)
            C(i, j) = S
        Next j
    Next i
    For i=1 To m
        t = 2
        For j = 1 To l
            Форма1.Print Tab(t); C(i, j);: t = t + 7
        Next j
    Форма1.Print
    Next i
End Sub
```

У меню **Разработать** вибрати команду **Добавить модуль.**

У вікні **Add Module** обрати **Новый** і натиснути кнопку **Открыть.**

З'явиться вікно Проект1 – Модуль1 (Код). Надрукуємо ім'я процедури Sub Form(k,f1,f2,z!()) і натиснемо **Enter**. З'явиться End Sub (кінець процедури).

Після заголовку процедури набрати текст(тіло процедури).

```
Private Sub Form(k, l, f1, f2, z!())
    For u = 1 To k
        t = 2
        For q = 1 To l
            z(u, q) = Int((f2 - f1 + 1) * Rnd) + f1
    Форма1.Print Tab(t); z(u, q);: t = t + 7
Next q
```

В цій процедурі елемент матриці створюється за допомогою генератора випадкових чисел. Для введення реальної конкретної матриці досить замінити оператор

z(i,j)=int((f2-f1+1)*Rnd)+f1 на

z(i,j)=Val(InputBox("Введіть елемент матриці").

В меню **Разработать** вибрати команду **Добавить модуль.**

В вікні **Add Module** вибрати **Новый** і натиснути кнопку **Открыть.**

З'явиться вікно Проект1 – Модуль2 (Код). Надрукуємо ім'я процедури Sub Sum (k,z!(),s!) і натиснемо **Enter**. З'явиться End Sub (кінець процедури).

Після заголовку процедури набрати текст(тіло процедури).

```
Sub Sum(i, j, n, S)
S = 0
    For k = 1 To n
        S = S + a(i, k) * b(k, j)
    Next k
End Sub
```

12. Запустити команду на виконання, натиснувши кнопку **Начать** на панелі інструментів **Стандартная**.

13.У вікні Форма1 клацнути по командній кнопці Пуск. Ввести початкові дані m,n,L,f1,f2.

Результат виконання процедури буде показаний у вікні форми (рис.6.2).

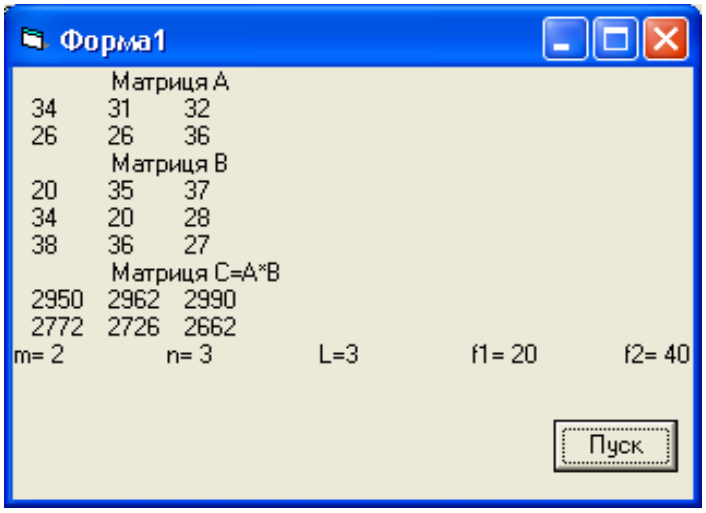



**Рисунок 6.2.** Результат знаходження добутку матриць

15. Для закінчення розрахунків необхідно клацнути по кнопці **Конец** на панелі інструментів.

16. Зберегти проект, клацнувши **Сохранить** (**Save Project**).

**Приклад**. *Створити додаток для обчислення ймовірності події за алгоритмом, блок-схема якого відображена на рис. 1.10.*

1. Відкриваємо новий проект.

2. На формі встановимо 5 елементів TextBox.

3. Відкриваючи вікна їх властивостей, видалимо з вікон текст по замовчуванню (Текст1, Текст2,…). Поруч з текстовими вікнами розмістимо елементи Label з написами про призначення тестових вікон. Подвійним щигликом на вікні елемента Текст4 відкриємо вікно коду з рядками.

```
Private Sub Текст4_Exchange()

End Sub
```

4. Замінимо подію Exchange на LostFocus і введемо наступний код головної процедури

```
Private Sub Текст4_LostFocus()
Dim u!, v!, R1!, R2!, R3!, p!, no%, n%, m%, k%, D!
    no = Текст1.Text
    n = Текст2.Text
    m = Текст3.Text
    k = Текст4.Text
    Call R(n, k, R1)
    Call R(no - n, m - k, R2)
    Call R(no, m, R3)
    D = R1 * R2 / R3
    Текст5.Text = D
End Sub
```

5. Потім введемо код підпрограми R(u,v,R):

```
Private Sub R(u, v, R)
    Dim C1!, C2!, C3!
    Call Fakt(u, C1)
    Call Fakt(v, C2)
    Call Fakt(u - v, C3)
    R = C1 / C2 / C3
End Sub
```

6. Після цього або перед цим (порядок розміщення процедур не має значення) вводимо код підпрограми Fakt(z,p).

```
Private Sub Fakt(z, p)
p = 1
    For i = 1 To z
        p = p * i
    Next i
End Sub
```

7. Клацнемо кнопку **Начать** на панелі інструментів.

8. Введемо в текстові поля початкові дані. Четверте вікно заповнюємо останнім і клацаємо на п’ятому. При цьому четверте вікно втрачає фокус і починає виконуватись головна процедура, яка відповідає такій події. В п’ятому вікні з’явиться результат (рис.6.3)

У Basic *N* і *n* не відрізняються, тому замість імені змінної N використовуємо ім’я $n_o$.

Коли головна процедура звертається до підпрограми R(u,v,R), наприклад, за командою Call R(k,n,R1), то формальні параметри заміщуються фактичними, так

ніби u=k, v=n, а результат R1=R. За наступною командою Call(no-n, m-k, R2) u=no-n, v=m-k, а результатом стане R2 і т.д. В свою чергу процедура R при кожному звертанні до неї сама тричі звертається до підпрограми Fakt(z,p), Це здійснюється послідовністю операторів

Call Fakt(u, C1)

Call Fakt(v, C2)

Call Fakt(u - v, C3)

В наслідок виконання яких, отримуємо C1=u!, C2=v!, C3=(u-v)!

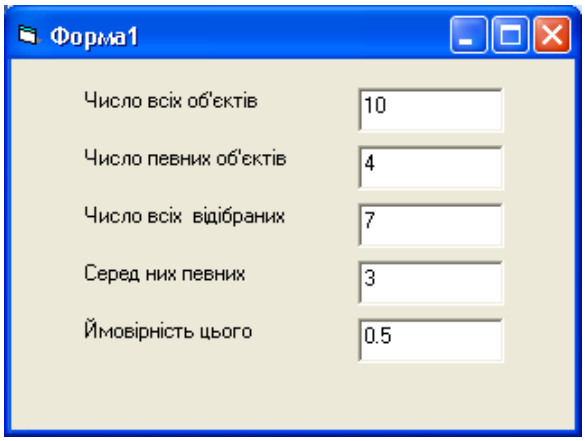



**Рисунок 6.3.** Результат обчислення ймовірності.

**Приклад.** *Обчислити наближене значення функції Лапласа для* $0< x <5$ *за формулою* $S=\sum_{i}^{\infty}\frac{(-1)^n\cdot x^{2i+1}}{\sqrt{2\pi}\cdot(2i+1)\cdot 2^i\cdot i!}$ *з точністю до 0.00001.*

Скористаємось алгоритмом розглянутим в розділі 1 (рис.1.10). На формі розмістимо лише один елемент управління – командну кнопку, якій дамо ім’я Обчислити. Значення x будемо вводити за допомогою функції InputBox, а результат виводити безпосередньо на форму за допомогою оператора Print.

```
Private Sub Обчислити_Click()
Dim Fi As Single, x As Single, eps As Single, i As Integer
    s = 0
    i = 0: eps = 0.00001
    x = Val(InputBox("x=", " Ввести значення  x"))
    Do
    a = (-1) ^ i * x ^ (2 * i + 1) / (2 * 3.14159) ^ 0.5 / ((2 * i + 1) * 2 ^ i * Fact(i))
    i = i + 1
    s = s + a
    Loop Until (Abs(a) < eps)
    Print x, Format(s, "#.#####")
End Sub
```

```
Function Fact(i)
P = 1
    For t = 1 To i
        P = P * t
    Next t
Fact = P
End Function
```

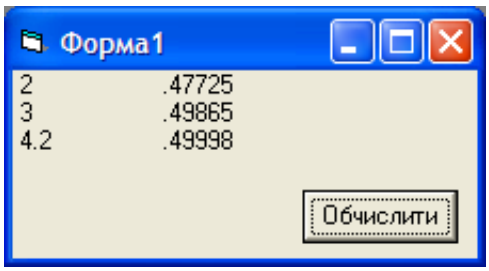



**Рисунок 6.4.** Результат обчислення функції Лапласа

**Приклад.** *Створити за допомогою генератора випадкових чисел Матрицю A. Показати її у вікні Текст1. Впорядкувати її за зростанням. Знайти суму позитивних елементів матриці.*

1. Відкрити новий проект. На формі встановити командну кнопку Пуск і два елементи TextBox з розмірами достатніми для відображення матриць. Суму позитивних елементів передбачається вивести у вікні MsgBox.

2. У вікнах властивостей текстових вікон очистити властивості Текст, а властивостям Multiline надати значення Да.

В вікні коду, що виникне після подвійного щиглика на кнопці Пуск записати наступну програму.

```
Option Base 1
Private Sub  Пуск_Click()
Dim A(), B(), S!, m%, n%, i%, j%, t!
    m = Val(InputBox("m=", Число рядків  m", "Ввести"))
    n = Val(InputBox("n=", "Число стовпців  n", "Ввести"))
    ReDim A(m, n), B(m, n)
    Текст1.Text = ""
    For i = 1 To m
        For j = 1 To n
            A(i, j) = 20 * Rnd - 10
            Текст1.Text = Текст1.Text & "   " & CSng(A(i, j))
        Next j: Текст1.Text = Текст1.Text & Chr(13) & Chr(10)
    Next i
    Текст1.Text = Текст1.Text & "   До впорядкування"
    For i = 1 To m
```

```
        For j = 1 To n
            For u = 1 To m
                For v = 1 To n
                    If A(i, j) < A(u, v) Then
                    t = A(i, j):   A(i, j) = A(u, v):   A(u, v) = t
                    End If
                Next v
            Next u
        Next j
    Next i
    Текст2.Text = ""
    For i = 1 To m
        For j = 1 To n
            Текст2.Text = Текст2.Text & "   " & CSng(A(i, j))
        Next j: Текст2.Text = Текст2.Text & Chr(13) & Chr(10)
    Next i
    S = 0
    For i = 1 To m
        For j = 1 To n
            If A(i, j) > 0 Then S = S + A(i, j)
        Next j, i
    Текст2.Text = Текст2.Text & "  Після впорядкування"
    MsgBox "Сума позитивних елементів  S=" & S
End Sub
```

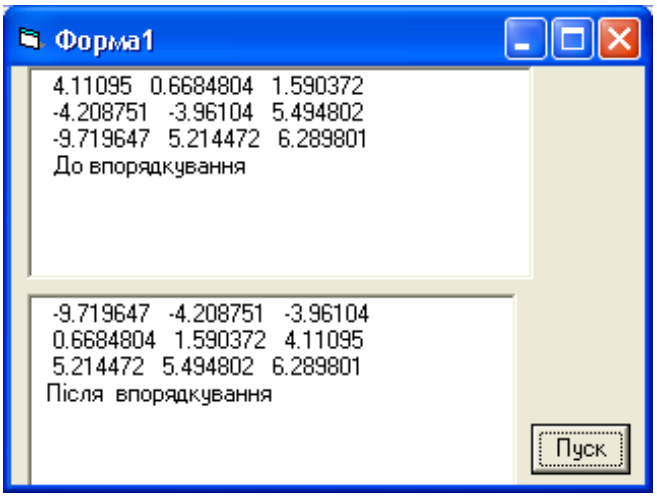



**Рисунок 6.5.** Впорядкування елементів матриці.

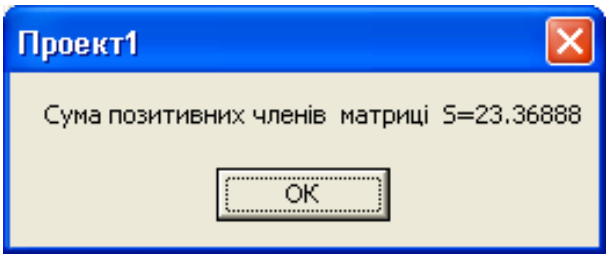



**Рисунок 6.6.** Результат обчислення суми елементів

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Розкрийте поняття модульна структура програми.
2. Які типи процедур ви знаєте?
3. Які дії потрібно виконати щоб додати до проекту процедуру визначеного типу?
4. В яких випадках виникає необхідність використання процедури Function…End Function? Наведіть приклади.
5. Звернення до підпрограми Function…End Function. Які аргументи використовуються при звернені до цієї підпрограми?
6. В яких випадках виникає необхідність використання процедури Sub…End Sub? Наведіть приклади.
7. Які параметри використовуються у процедурі Sub…End Sub?
8. Що необхідно зробити, щоб передати до процедури масив?
9. Що таке фактичний аргумент?
10. Що таке лінійна форма функції користувача? Який вона має формат?

# *Лекція 7*

# РОБОТА З ФАЙЛАМИ

***План лекції:***

*7.1. Типи доступу до файлів*

*7.2. Обробка файлових структур даних з послідовним доступом*

*7.2. Обробка файлових структур даних з довільним доступом*

Основним інформаційним об'єктом у ПК є файл. Як тип даних *файл* – це іменована послідовність однорідних компонентів, що зберігаються на магнітних дисках. Кількість записів у файлі чітко не фіксується і може змінюватися. Будь-який файл можна розглядати як доступну область зовнішньої пам'яті з певним ім'ям, в якій зберігається деяка сукупність даних.

Файли даних на магнітних носіях інформації дозволяють практично необмежено довго зберігати вхідні та вихідні дані програми.

## 7.1. ТИПИ ДОСТУПУ ДО ФАЙЛІВ

Тип даних, що містяться у файлі, визначає тип доступу до нього. У Visual BASIC 6.0 реалізовано три типи доступу до файлів:

- послідовний – для читання та запису текстових файлів;
- довільний – для читання і запису тексту або структурованих двійкових файлів із записами фіксованої довжини;
- двійковий – для читання та запису довільно структурованих файлів.

З вищеперерахованих режимів доступу тепер найбільш часто використовується послідовний, оскільки в практиці програмування нерідко виникає необхідність запису даних втекстовий файл, для чого послідовній доступ найбільш зручний. Значно рідше в наш час находять застосування довільний і двійковий режими доступу, які достались Visual Basic у спадок з тих часів, коли ще не було Windows з її можливостями і програмісту доводилось самому писати підпрограми для здійснення складних дискових операцій з файлами баз даних, малюнками і іншими складними форматами. В наш час це здійснюється значно простіше з використанням обєктів і методів інших додатків Windows. Операційна система або будь-який додаток, у тому числі розроблений мовою програмування Visual BASIC 6.0 зв'язується з файлом за допомогою каналу введення-виведення. Під час відкривання файлу ставиться у відповідність канал за допомогою якого записуються або прочитуються дані.

Процес відкривання і збереження файлів складається з кількох етапів:

- відкриття файлу;
- читання або запис даних;
- закриття файлу.

## 7.2 ОБРОБКА ФАЙЛОВИХ СТРУКТУР ДАНИХ З ПОСЛІДОВНИМ ДОСТУПОМ

Послідовний доступ розміщує елементи (записи) у файлі за принципом "наступний після попереднього". У такій послідовності елементи з файлу і читаються – спочатку перший, потім другий, далі третій і т.д. Двадцять перший елемент можна прочитати тільки після двадцятого. Послідовний доступ застосовується головним чином для роботи з текстовими файлами, тобто з файлами, елементи яких записано у вигляді символів. Кожен елемент може мати довільну довжину. Тому вони розподіляються спеціальним символом. Дії в режимі послідовного доступу подібні до роботи з аудіозаписами на касеті в магнітофоні.

Послідовний доступ краще використовувати для файлів, що складаються тільки з тексту, створених за допомогою типового текстового редактора, в яких дані не поділяються на послідовність записів. Послідовній доступ не дуже підходить для збереження довгого ряду чисел, оскільки кожне число у послідовному файлі зберігається як символьний рядок. У цьому разі для збереження чотиризначного цілого числа були б потрібні 4 байти замість 2.

**Відкриття файлів для послідовного доступу**

Щоб відкрити файл для послідовного доступу, потрібно використати такий синтаксис оператора Open:

*Open Ім'яФайла For <Режим_роботи> As #нф*

Ім'я файлу – це або рядок символів, взятий у лапки, або вираз, значення якого є рядок символів. Він представляє собою шлях (маршрут) до файлу, що відкривається. Якщо вказано тільки ім'я файла, файл повинен розташовуватись у поточній папці.

<Режим роботи> – це одне з трьох ключових слів: **Output, Append, Input.**

*Output* – якщо файл відкривається для запису до нього даних, починаючи з першої позиції.

*Append* – якщо файл відкривається для запису до нього даних не з першої позиції, а з кінцевої.

*Input* – якщо файл відкривається для читання з нього текстових даних.

нф – номер (дескриптор) файла – будь-яке число від 1 до 511. Воно потрібно для ідентифікації файла в програмі.

Якщо файл не існує і відкривається для читання (For Input), то Visual Basic видає повідомлення про помилку, а якщо для запису або додання (Output чи Append), то створюється новий файл. Якщо файл з вказаним ім'ям існує, то в режимі Output його вміст вилучається, а в режимі Append файл відкривається для додавання символів:

Open "C:\README.TXT" For Input As #1

Open "C:\DATA\TEXT.TXT" For Output As #2

Open "C:\USERS.TXT" For Append As #3

Після відкривання файла для виконання операцій Input, Output або Append його треба закрити за допомогою оператора Close, перш ніж знову відкрити для виконання операції іншого типу.

**Закриття файлів**

Всі відкриті текстові файли закриваються однаково за допомогою оператора *Close #[<Список_ Дескрипторів>]*.

<Список_Дескрипторів> – це записані через кому ідентифікатори файлів, які повинні бути закрити. Якщо <список дескрипторів> відсутен, будуть закриті всі відкриті файли.

**Запис у файл**

У Visual Basic для запису інформації у файл використовуються оператори *Print #* та *Write #.*

Синтаксис операторів запису в текстовий файл однаков:

*Print # нф, <Список_ Значень> Write #нф, <Список_ Значень>*

нф – це ціле число, яке повинно збігатись з ідентифікатором відкритого для запису файла;

<Список_Значень> – це записані через розподільник значення (або змінні). Якщо <Список_Значень> відсутній, то в файл буде записан порожній рядок.

Для форматування інформації, що записується у файл потрібно по-різному відокремлювати дані в операторі Print. Якщо їх відокремлювати комами, то у файлі вони будуть відокремлені символами табуляції.

Якщо ж в операторі для відокремлення даних використати крапку з комою(;), то дані у файл записуються без роздільників. Крім того, в <Списку Значень> оператора Print можуть бути включені функції:

Spc(n) – для вставки n пробілів між значеннями в текстовому рядку;

Tab(n) – для вказівки номера n позиції для запису наступного значення.

Розподільником в <Списку Значень> в операторі Write # є кома. Список значень переглядається послідовно, та елементи цього списку записуються в один текстовий рядок файла через кому. Елементи типу String заключаються в лапки. Після запису останнього елемента записується символ переходу на новий рядок.

Якщо Print # зберігає дані у вигляді звичайного тексту, то Write # форматує текстові рядки в лапки а цифри виводяться без лапок. Наприклад:

```
Print # 1, "Київ"; "Харків"; 25          ' у файлі буде: Київ Харків 25
Write # 2, "Київ", "Харків"; 25          ' у файлі буде "Київ"; "Харків"; 25
```

Оператор Print зручен для охайного редагування тексту вихідного файла. Оператор Write краще застосовувати, коли вихідний файл буде використовуватись надалі як вхідний для інших програм.

**Редагування файлів послідовного доступу.**

Щоб відредагувати файл послідовного доступу, спочатку треба ввести записи з файла у програмні змінні, після чого змінити їх і записати знову у файл.

**Читання з файла.**

Читання даних з файла, відкритого для послідовного доступу, здійснюється за допомогою оператора Input, що має кілька різновидів:

Input # – прочитує послідовність символів, записаних за допомогою оператора Write #;

Line Input # – прочитує один рядок;

Input$ – прочитує певну кількість символів.

Перед читанням треба відкрити файл за допомогою оператора *Open...For*.

Наприклад:

```
Open "C:\Text.Txt" For Input As #1
```

Оператор Input має наступний синтаксис:

*Input #нф, <Список Змінних>*

нф – це ціле число, яке повинно збігатись з ідентифікатором відкритого для читання файлу;

<Список_Змінних> – це записані через кому змінні. В кожному текстовому рядку файла кількість та тип змінних повинно збігатись з кількістю та типом значень в <Списку Значень> оператора Write.

Оператор Line Input має наступний синтаксис:

*Line Input # нф, <Змінна>*

*Змінна* – змінна типу String або Variant. Результатом роботи оператора Line Input є присвоєння <Змінній> значення всього текстового рядка файла.

Читання із текстового файла виконується звичайно циклічно за допомогою оператора циклу з умовою *Do While...Loop* або *Do Until...Loop*. Умовою закінчення циклу є спроба прочитати дані після читання останнього текстового рядка. Ця спроба приводиіь до того, що після досягнення кінця файла значення функції EOF(нф) буде True.

**Функція Input$** – це функція двох аргументів:

*Input$(Кількість_Символів, нф)*

Перший її аргумент – це кількість символів, які треба прочитати із вхідного файла.

Другий аргумент – ідентифікатор файла, відкритого для читання.

Повертаєме значення – прочитаний текст в вигляді символьного рядка.

Цю функцію використовують для одночасного читання всього текстового файла та розміщення його в текстовому полі екранної форми. Для цього необхідно визначити довжину файла в байтах за допомогою функції LOF(нф).

Наприклад, для читання всієї інформації з файла можна запропонувати один з двох варіантів:

Варіант 1

```
Do Until EOF (1)
Line Input #1, String
Text= Text & String
Loop
```

Варіант 2

```
Text=Input$(LOF(1),1)
Close #1
```

Обидва варіанти приводять до однакового результату.

**Приклад** створення і використання файла з послідовним доступом.

*Продавцям необхідно зберегти відомості про продані товари, наприклад, в цілях обліку та для підбиття підсумків в кінці дня.*

Використаємо для рішення цієї задачі файл з послідовним доступом. Розмістимо на Формі 1, яка використовувалась в додатку «Покупки» ще три командні кнопки, яким дамо імена Записати, Доповнити, Показати та змінимо відповідно написи на них.

Щигликам на цих кнопках ставимо їм в відповідність процедури Записати_Click(), Доповнити_Click, Показати _Click. Відображати вміст файлу з записами можна було б на окрему форму, додану з цією ціллю до проекту, але якщо записів багато, то вони можуть на ній не розміститись. Тому повний список прода-

ного товару будем виводити в Список4, в якому при потребі виникне смуга прокручування.

```
Private Sub Обчислити_Click()
Dim Код As Integer, Ціна As Currency, S As Single, i As Integer
    S=0: ‘ Загальна вартість
    i=0 ‘ номер покупки
    Do While MsgBox(“Будуть ще покупки?”,vbQuestion+_
                        vbYesNo, “Покупки”)=vbYes
        i=i+1
        Код=Val(InputBox(“Введіть ціну”, i & “ –го товару”))
        Ціна=CCur(InputBox(“Введіть ціну”, i & “ –го товару”))
        Кількість=Val(InputBox(“Введіть ціну”, i & “ –го товару”))
        Список1. AddItem Код
        Список2. AddItem   Ціна
        Список3. AddItem   Кількість
        S=S+Ціна*Кількість
    Loop
    MsgBox “Вартість всіх покупок “  & Format( S, “0.00”) & “Грн.”
End Sub

Private Sub Записати_Click()
      Open "D:\Продажі.txt" For Output As #1
      For i = 0 To Список1.ListCount - 1
          Write #1, Список1.List(i), Список2.List(i), Список3.List(i)
      Next i
Close #1
End Sub

Private Sub Додати_Click()
      Open "D:\Продажі.txt" For Append As #1
      For i = 0 To Список1.ListCount - 1
          Write #1, Список1.List(i), Список2.List(i), Список3.List(i)
      Next i
Close #1
End Sub

Private Sub Показати_Click()
Dim Код As String * 4, Ціна As String * 15
Dim Кількість As String * 15
      Open "D:\Продажі.txt" For Input As #1
      S = 0
      Do While Not EOF(1)
          Input #1, Код, Ціна, Кількість
```

```
            Список4.AddItem Код & "          " & Ціна & "          " & Кількість
            S = S +Val( Ціна) *Val( Кількість)
        Loop
Close #1
Список4.AddItem ""
Список4.AddItem ""
Список4.AddItem "Всього на " & S & " Грн"
End Sub
```

Для того, щоб змінні, які відображуються в списку 4 мали однакову довжину, що необхідно для вирівнювання інформації в стовпчиках при її відображені, ми використали оператор *Dim Ім'я As String*N*. Він вказує, що VB повинен створити рядкову змінну фіксованої довжини (довжиною N символів).

На рисунках 7.1 та 7.2 відображені вікна з результатами роботи додатка.

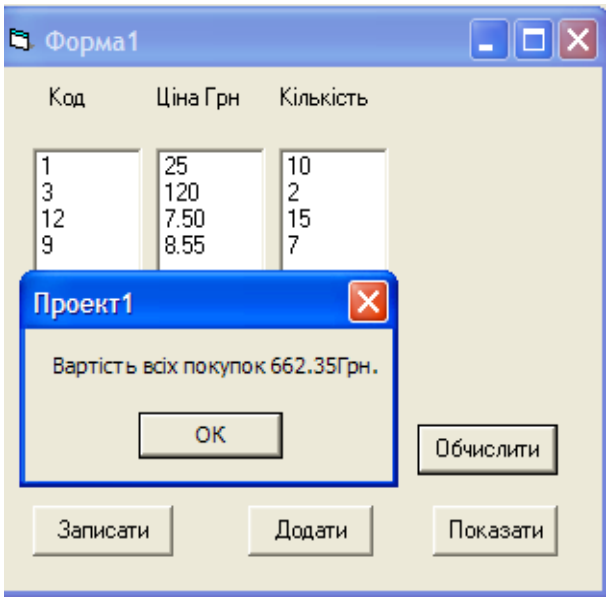



**Рисунок 7.1.** Результат роботи процедури "Обчислити_Click

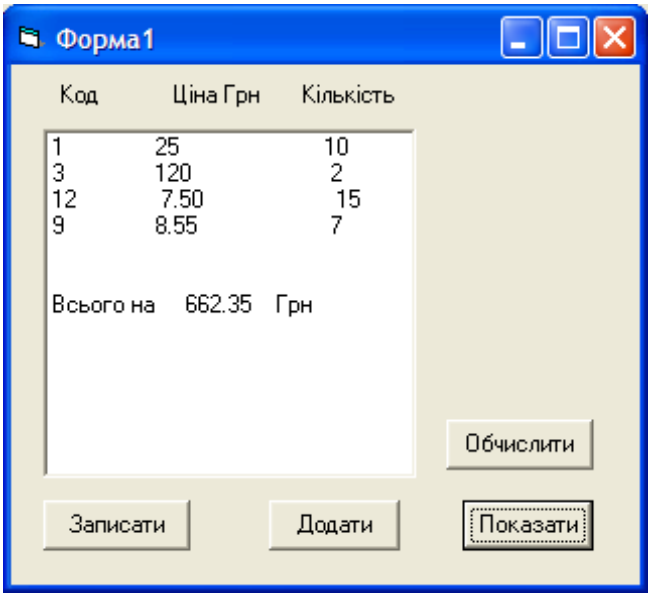



**Рисунок 7.2.** Результат роботи процедури "Показати_ Click"

## 7.3. ОБРОБКА ФАЙЛОВИХ СТРУКТУР ДАНИХ З ДОВІЛЬНИМ ДОСТУПОМ

**Відкриття файлу з довільним доступом**

Для роботи з файлом у режимі довільного доступу його потрібно відкрити оператором Open, що має вигляд:

*Open Ім'яФайла For Random As [#] Дескриптор [Len=ДожинаЗапису]*

де Ім'яФайла – вираз рядкового типу, що подає ім'я файлу. Наприклад:

"D:\ТК17\Файли_даних\Ціна.dat"

"Ціна.dat"

У першому випадку зазначено папку, в якій зберігається файл Ціна.dat, а в другому передбачається, що він зберігається у поточній папці.

Нижче наведено деякі поняття, що стосуються роботи з файлами з довільним доступом.

*Дескриптор* – вираз цілого типу, що визначає номер каналу введення/виведення для файлу, що відкривається (наприклад, 1). Символ # перед дескриптором необов'язковий.

Довжина запису – вираз цілого типу, що визначає розмір елемента в байтах (наприклад, 25). Часто для завдання довжини запису використовують вмонтовану функцію Len(*x*), що визначає розмір аргументу *x*. При відкритті файлу *x* вказує ім'я змінної типу даних користувача, що буде вживатися для роботи з файлом.

```
Open txtІм'яФайла For Random As # 1 Len (udtВідомості)
```

Відкривається файл, ім'я якого зазначено в текстовому полі txtІм'яФайла. Довжина записів у файлі збігається з розміром змінної типу даних користувача udtВідомості. Для роботи з файлом призначений перший канал введення/виведення.

**Запис у файл**

Для записування даних у файл використовують оператор Put, для читання – оператор Get. Ці оператори мають вигляд

*Put # дескриптор, Номер запису, Змінна*

*Get # дескриптор, Номер запису, Змінна*

Діє оператор Put так: значення, що зберігається в полі пам'яті змінної, пересилається з оперативної пам'яті у файл, обумовлений дескриптором, і у файлі записується на тім місці, яке задано номером запису.

Оператор Get має протилежне призначення. Він вказує, що потрібно знайти запис із зазначеним номером у файлі, що визначений дескриптором, а потім переслати вміст цього запису з файлу в поле, що займає зазначена змінна в оперативній пам'яті.

```
Put # 1, i, udtВідомості
Get # 1, i, udtВідомості
```

У першому випадку здійснюється запис у файл, пов'язаний із першим каналом введення/виведення, інформації, що зберігається в оперативній пам'яті в змінній типу даних користувача udtВідомості. Ця інформація записується у файл на місце з номером *i*.

У другому випадку відбувається зворотний процес: з і-го місця у файлі зчитується інформація і пересилається в оперативну пам'ять у змінну udtВідомості.

Для закриття файлів використовують оператор Close, що має вигляд:

*Close [#] [СписокДескрипторів]*

Якщо списку дескрипторів немає, то закриваються усі відкриті файли.

```
Close # 1,2,5
Close
```

У першому випадку закриваються тільки файли, для яких було призначено канали введення/виведення 1, 2 і 5, а другому – усі файли.

**Перейменування, копіювання, видалення файлів з довільним доступом**

Перейменування файлу здійснюється оператором Name, що має вигляд:

*NameСтареІм'я As НовеІм'я*

Копіюють файли за допомогою оператора FileCopy, що має вигляд:

*FileCopy Ім'яВихідногоФайла, Ім'яКінцевогоФайла*

Щоб видалити файл із певним ім'ям, застосовують оператор Kill

*Kill Ім'яФайла*

У всіх цих операторах імена файлів задаються рядковими виразами, що визначають ім'я файла і, можливо, шлях до нього.

Приклади операторів перейменування, копіювання і видалення файлів:

```
Name "Лютий.dat" As "Березень.dat"
FileCopy "Податки.dat", "Утримання.dat"
Kill "Податки.dat"
```

У першому випадку файл із даними за лютий перейменуються на файл із даними за березень. У другому – дані з файлу про податки копіюються в новий файл, що містить відомості про утримання. У третьому вилучається файл із даними про податки.

Оскільки всі записи у файлі з довільним доступом мають однакову довжину, то, знаючи загальну довжину всього файлу, можна визначити кількість записів у ньому за допомогою такого оператора:

*IntКільк_зап =Lof(x)/Len(udtВідомості)*

Тут Lof(x) – вмонтована функція VB, за допомогою якої визначається загальна довжина файлу, пов'язаного з каналом x (Lof – Length Of File – довжина файлу).

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Поясніть, що означають наступні терміни: файл, запис, метод доступу, структура запису?
2. Порівняйте поняття «Файл» та «Канал введення/виведення».
3. Яке призначення операторів відкриття та закриття файлів?
4. У яких випадках використовується тип даних користувача? Наведіть приклади.
5. Якими операторами описується тип даних користувача? Наведіть приклади.
6. Чи допустимі різні типи даних для елементів одного запису?
7. У яких випадках використовуються файли та які операції можна використовувати з ними? Наведіть приклади
8. Вкажіть, за допомогою яких операторів виконується запис даних у файл послідовного доступу (довільного доступу)?
9. Вкажіть, за допомогою яких операторів виконується запис даних у файл довільного доступу?
10. Якими операторами передаються дані між пам’яттю і файлом у режимі довільного доступу?
11. Вкажіть, за допомогою яких операторів виконується читання із файлу?
12. Як розпізнати кінець файлу даних?
13. Як розпізнати файл на диску?
14. Як розпізнати довжину файлу?
15. У яких випадках закривають файл?