МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
АВТОМОБІЛЬНО-ДОРОЖНИЙ УНІВЕРСИТЕТ

ЛЕКЦІЇ З ДИСЦИПЛІНИ

"ІНФОРМАТИКА І ПРОГРАМУВАННЯ"

За напрямом підготовки 6.080101 Геодезія, картографія та землеустрій Розділ «Основи програмування в середовищі мови Visual Basic»

Розробник
Доцент кафедри ІТ та М
Кудін А.І.

Харків ХНАДУ 2016

ЗМІСТ

Лекція №1

Тема: АЛГОРИТМІЗАЦІЯ ОБЧИСЛЮВАЛЬНОГО ПРОЦЕСУ

Мета лекції: вивчити основні поняття алгоритмізації обчислювального процесу, типів і способів опису алгоритмів, придбання навиків в складанні графічних схем алгоритмів.

Час: 2 години

Місце проведення: Аудиторія № Д, А

План

Вступ
1.1 Поняття алгоритму і його властивості
1.2 Типи алгоритмів
1.3 Поняття змінної. Присвоювання значень змінним
1.4 Способи опису алгоритмів
1.5 Основні блоки схем алгоритмів
1.6 Програма і алгоритм
1.7 Етапи рішення задач на ЕОМ
1.8 Контрольні питання

м. Харків
2016р.

Вступ

Поняття алгоритму в його широкому значенні належить до числа основних понять інформатики і кібернетики.

Своїм походженням слово алгоритм, спочатку воно звучало як "алгорифм", зобов'язано давньогрецькому математику Евкліду (3 в. до н.е.) який назвав так на честь узбецького математика Аль-Хорезмі, який сформулював послідовність дій (правил) знаходження найбільшого загального дільника двох чисел.

Аль Хорезмі узбецький математик (787-860р. до н.е.) сформулював правила виконання чотирьох арифметичних дій для багатозначних чисел. Таким чином можна зробити висновок, що вперше поняття алгоритму введене математиками за довго до появи обчислювальної техніки.

1.1 Поняття алгоритму і його властивості

В даний час існує багато визначень терміну алгоритм. Наприклад, академік А.М. Колмогоров давав таке визначення: алгоритм – це всяка система обчислень, що виконується за строго певними правилам, яка після якогось числа кроків приведе до рішення поставленої задачі.

В інженерній практиці часто використовується таке визначення: алгоритм – це кінцева послідовність певних дій, які визначають процес переходу від вихідних даних до рішення поставленої задачі.

Ми будемо використовувати таке визначення.

Алгоритм – зрозуміле і точне розпорядження виконавцю зробити послідовність дій, направлену на досягнення поставленої мети або рішення поставленої задачі.

Наприклад, порядок дій водія при початку руху автомобіля: зняти автомобіль з ручного гальма, натиснути на педаль зчеплення, ввімкнути першу передачу, поступово відпускати педаль зчеплення, натискаючи на педаль газу.

Виходячи з визначення і наведеного прикладу, ми можемо зробити висновок про те, що поняття алгоритму охоплює дуже багато областей діяльності людини. Наприклад: порядок проїзду від

південного вокзалу до університету, порядок користування телефоном і таке інше.

В результаті розвитку теорії алгоритмів було визначено, що будь-який алгоритм повинен мати наступні властивості:

Дискретність – алгоритм повинен бути послідовністю виконання простих, наперед визначених кроків. Для виконання кожного кроку алгоритму потрібен певний відрізок часу, тобто отримання результату на підставі початкових даних відбувається дискретно.

Визначеність – алгоритм абсолютно точний, однозначний і зрозумілий для виконавця. Його неодноразове застосування до одних і тих же початкових умов приводить до одних і тих же результатів.

Формальність – процес виконання алгоритму носить механічний характер, що дозволяє застосовувати для виконання алгоритму різні машини і механізми.

Результативність – при виконанні алгоритму обов'язково виходить передбачений результат за певну кінцеву кількість кроків.

Масовість – можливість застосування конкретного алгоритму при рішенні багатьох задач даного типу.

Висновок: при складанні алгоритму необхідно перевіряти, чи володіє він усіма вище перерахованими властивостями.

1.2 Типи алгоритмів

Алгоритми можуть бути дуже складними і мати опис дуже багатьох факторів. Наприклад, алгоритм опису зльоту літака. Можуть бути дуже простими. Наприклад, проїзд від вокзалу до університету, використання телефону-автомата та ін. Звичайно складні алгоритми можна розбити на велику кількість простих алгоритмів.

Прості алгоритми – це алгоритми, що складаються з невеликої кількості елементарних дій. Умовно їх розподілили на такі типи: **лінійні, розгалужені і циклічні**.

Лінійний алгоритм – це алгоритм, у якого всі дії виконуються послідовно одна за одною. При цьому всі дії виконуються тільки один раз.

Приклад: проїзд від вокзалу до університету.

Розгалужені алгоритми – це алгоритми, в яких крім послідовності дій є операції умови. Умова в них поставлена в вигляді питання таким чином, що на нього можна відповісти однозначно так чи ні.

В таких алгоритмах дії виконуються послідовно одна за одною до операції умови. При виконанні операції умови аналізується поставлена умова. І якщо умова виконується, тобто на поставлене питання можна відповісти «так», то далі виконуються дії записані по гільці (напрямку) «так». Якщо умова не виконується, то виконуються дії, записані по гільці «ні».

Циклічні алгоритми – це алгоритми, у яких одна і та сама послідовність дій виконується кілька разів, причому при кожному новому виконанні цієї послідовності змінюється один чи декілька вихідних даних (параметрів) і кінцевий результат.

Циклічні алгоритми поділяються на **арифметичні та ітераційні.**

Арифметичні – це алгоритми, у яких заздалегідь відома або може бути вирахувана кількість повторень. Наприклад, налити десятилітровий посуд літровим кухлем.

Ітераційні алгоритми – це алгоритми, у яких дії виконуються до вказаного результату. Наприклад, накачати шину автомобіля до тиску 2 атм.

У чистому вигляді вказані типи алгоритмів зустрічаються дуже рідко. Звичайно вони складаються з набору перерахованих типів.

1.3 Поняття змінної. Присвоювання значень змінним

Поняття змінної є одним з основних понять в інформатиці і програмуванні. При використанні ЕОМ це трактується таким чином: під змінною розуміють ім'я комірки (клітки) пам'яті, куди можуть бути "записані" (поміщені) які-небудь дані.

При цьому, для того щоб виконати які-небудь дії або операції над даними, необхідно записати виконання цієї операції над ім'ям цієї комірки (клітки). У загальному випадку ім'я змінної може складатися з набору символів (букв, цифр), але першим символом повинна бути буква.

Довжина слова, яке є ім'ям змінної, може бути різною. Вона обмовляється при описі кожної мови програмування синтаксисом мови. У найпростішому випадку ім'я складається з однієї букви.

З поняттям імені змінної тісно пов'язане поняття привласнення значень змінним. У комірку пам'яті в різні моменти часу можуть бути поміщені різні дані тоді як ім'я комірки (клітки) не буде змінюватися.

Таким чином, привласнювання значення змінній – це процес поміщення даних в комірку (клітку) із вказаним ім'ям. Поміщення даних в комірки, називають привласнення значень коміркам. В алгоритмізації це записують так:

A=5,7

b = Zk

A=20,8

Це означає, що змінній A привласнено значення 5.7, потім 20.8, а змінній b – Zk, або, що те ж саме, в клітку A помістили 5.7, а потім 20.8, а в клітку b строкові дані Zk.

При записі дій над змінними правомірний наступний запис:

A=5

B=25,5

A=A+B

Такий запис означає: нове значення змінної A дорівнює старому значенню змінної A плюс значення змінної B. Нове значення змінної A буде 30,5.

При цьому необхідно пам'ятати: праву і ліву частину в такому виразі міняти місцями не можна. У лівій частині може бути тільки ім'я змінної і не може бути число чи арифметичний вираз.

У правій частині може бути, ім'я змінної, число або арифметичний вираз, але значення усіх змінних, які входять цей вираз на момент виконання операції повинні бути визначені, тобто їм повинні бути привласнені значення.

1.4 Способи опису алгоритмів

Розрізняють наступні способи опису алгоритмів.

Словесний – опис роботи алгоритму за допомогою тексту. Наприклад, інструкція користування телефоном-автоматом.

Математичний – опис за допомогою математичних формул. Наприклад:

S = V * T

W = a * b * h

При цьому повинно бути відомо, що означає кожен символ у формулі.

Графічний – опис за допомогою графічних схем і спеціальних геометричних фігур, які називаються блоками. Кожен блок означає певну дію. Призначення блоків і правила їх зображення закріплені державним стандартом (ГОСТ 19003-80, ГОСТ 19.701-90).

1.5 Основні блоки схем алгоритмів

Блоки початку та кінця алгоритму. Призначені для позначення початку і кінця алгоритму. Будь-який алгоритм починається з блоку ПОЧАТОК і закінчується блоком КІНЕЦЬ. Початок має один вихід, кінець – один вхід.

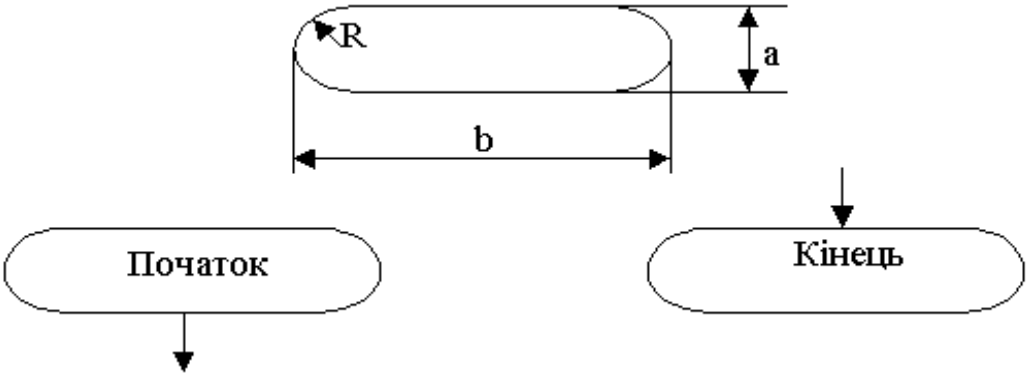


Рисунок 1.1 – Правила зображення та розміри блоків Початок та Кінець

Розмір «a» повинен вибиратися з ряду 10, 15, 20, 25 мм. і т. д. Розмір b = 2*a; R= a/2. Правила зображення приведені на рис. 1.1.

Блок вводу-виводу. Блок вводу призначений для привласнення значень змінним під час виконання програми. Блок виводу для виведення даних на відеомонітор або на друк під час виконання програми. Зображаються вони однаково у вигляді паралелограма з одним входом і одним виходом. В блоці вводу пишеться слово «Введення» і перелік змінних, яким необхідно привласнити значен-

ня. В блоці виводу пишеться слово «Виведення» і перелік змінних, значення яких необхідне вивести. Приклад зображення блоків приведений на рис. 1.2.

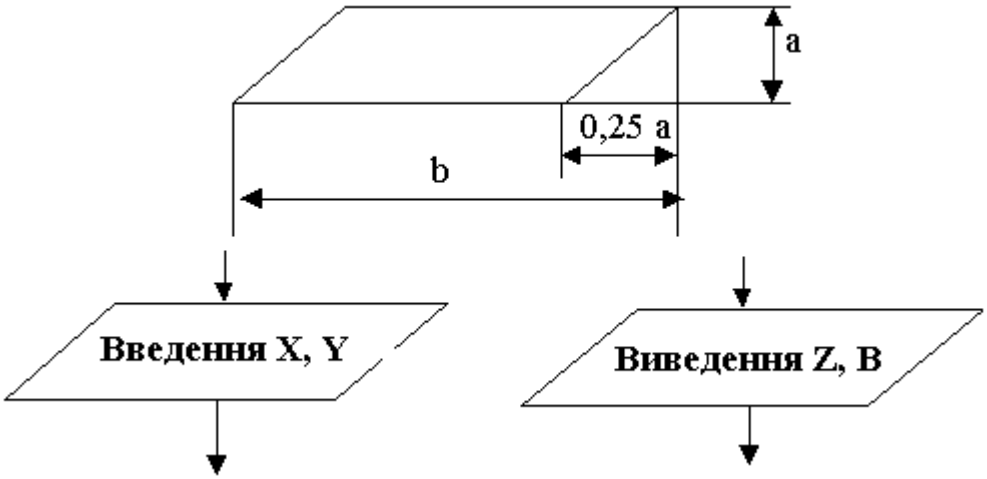


Рисунок 1.2 - Позначення блоків Введення та Виведення

Обчислювальний блок (блок процесу обчислень) призначений для запису математичних формул (значення усіх змінних, які розташовані в правій частині формули повинні бути визначені, тобто їм присвоєні конкретні значення). Приклад зображення обчислювального блоку показано на рис. 1.3, блок має один вхід і один вихід.

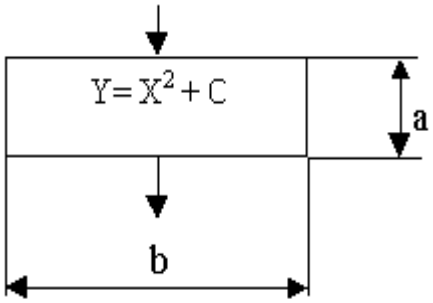


Рисунок 1.3 – Обчислювальний блок

Логічний блок (блок умови) призначений для реалізації розгалужених обчислювальних процесів. У цьому блоці записується умова, на яку можна відповісти тільки «та» чи «ні». Якщо умова виконується, то подальше виконання алгоритму йде по гілці «так», якщо не виконується – по гілці «ні». Блок має один вхід і два виходи. Зображення блоку показано на рис. 1.4.

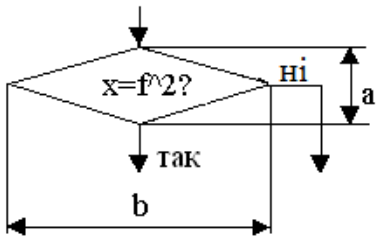


Рисунок 1.4 – Блок умови (розгалужень)

Блок модифікації (опису циклу) призначений для зображення арифметичного циклічного процесу. Розміри блоку вказані на рис.1.5 Блок має один зовнішній вхід і один зовнішній вихід, один внутрішній вихід на «тіло циклу» і внутрішній вхід від тіла циклу.

Параметри блоку модифікації: I – елемент, що управляє, який змінюється від початкового значення n до кінцевого значення k з кроком st. Якщо крок дорівнює 1, параметр st можна не указувати.

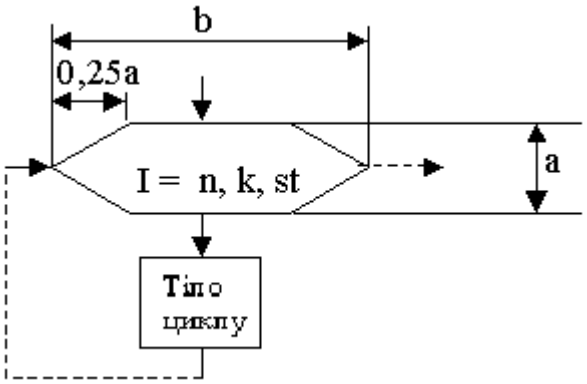


Рисунок 1.5 – Блок модифікації

Блок визначеного процесу призначений для позначення та використання раніше створених та вже описаних. Наприклад, знаходження мінімального елемента масиву. В середині блоку записується назва алгоритму. Приклад зображення на рис. 1.6.

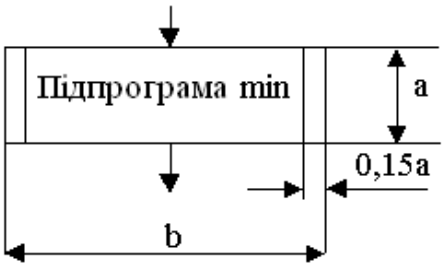


Рисунок 1.6 – Блок визначеного процесу

Блоки з'єднуються між собою суцільними прямими лініями із стрілками, які указують послідовність виконання дій при виконанні алгоритму. При записі обчислювальних блоків бажано, щоб в кожному блоці обчислювалося значення одного виразу. Такі блоки називаються операторами.

Розрив лінії зв'язку алгоритму для перенесення на іншу сторінку або продовження лінії зв'язку після розриву виконується у вигляді блоків розриву і продовження зв'язку (рис. 1.7).

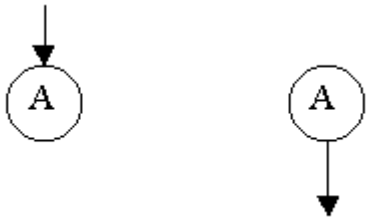


Рисунок 1.7 – Блок розриву зліва, блок продовження зв'язку справа

R=(1/2)*a

1.6 Програма і алгоритм

Розрив лінії зв'язку алгоритмуПрограма – це запис алгоритму на одній з мов програмування.

Відмінність між алгоритмом і програмою носить переважно контекстний або стилістичний характер. Звичайно говорять, що програма реалізує алгоритм, але не говорять, що програма є алгоритмом.

Програма, записана на мові програмування, в середовищі цієї мови перетворюється в машинні коди – мову, зрозумілу комп'ютеру. Це перетворення називається трансляцією. Існує два види трансляції.

Інтерпретація – переклад програми в машинні коди по одному рядку (по одному оператору) і одночасне їх виконання.

Компіляція – виконання програми здійснюється після того, як процес її перекладу повністю завершений.

1.7 Етапи рішення задач на ЕОМ

Рішення будь-якої задачі на ЕОМ складається з декількох етапів, основними з них є:

1) постановка задачі;

2) формалізація (математична постановка задачі);

3) вибір або розробка методу рішення;

4) розробка алгоритму;

5) складання програми;

6) відладка програми;

7) обчислення й обробка результатів (експлуатація програми).

Постановка задачі полягає у формулюванні кінцевої мети задачі, визначенні загального підходу до досліджуваної проблеми, з'ясуванні існування рішень поставленої задачі, визначенні переліку і розмірності початкових і вихідних даних. На цьому етапі потрібне глибоке розуміння поставленої задачі. Правильно сформулювати завдання іноді не менш складно, чим її вирішити.

Формалізація, як правило, зводиться до побудови математичної моделі розглянутого явища, процесу або розв'язуваної задачі. Вводиться система умовних позначок, визначається специфіка й обсяг вихідних даних.

Вибір методу рішення – побудова ряду формул і формулювання правил, що визначають зв'язки між цими формулами.

Розробка алгоритму – розкладання обчислювального процесу на можливі складові частини, установлення порядку їх виконання, опис змісту кожної такої частини в тій чи іншій формі.

Складання програми – запис алгоритму обраною мовою програмування.

Налагодження програми – знаходження і виправлення логічних і синтаксичних помилок. Перевірка роботи програми на контрольних прикладах.

Завдання 1. Ознайомитися із схемою алгоритму лінійного обчислювального процесу (рис. 1) для розв’язання задачі вирахування середньої швидкості руху автомобіля за формулою

V = l / t, де L - відстань; t – час руху автомобіля.

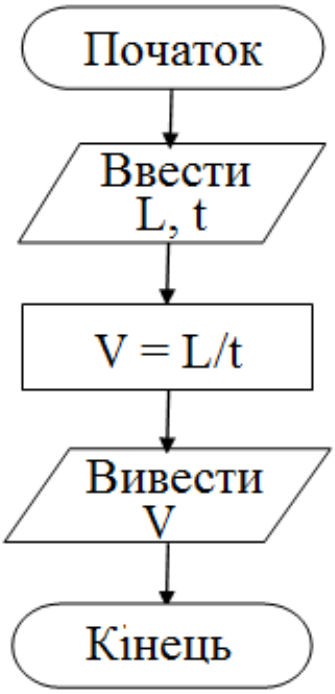


Рисунок 1.8 - Схема алгоритму лінійного обчислювального процесу

Контрольні запитання

1. Ким було введене поняття алгоритму?
2. Що таке алгоритм?
3. Якими властивостями повинен володіти алгоритм?
4. Які типи алгоритмів Ви можете виділити?
5. Що таке лінійний алгоритм?
6. Що таке розгалужений алгоритм?
7. Що таке циклічний алгоритм?
8. Що таке змінна?
9. Що таке ім’я змінної?
10. Які типи змінних Ви знаєте?
11. Які способи використовуються для опису алгоритмів?
12. Що таке графічний спосіб опису алгоритмів?
13. Які елементи використовуються для графічного способу опису алгоритмів, правила їх зображення та використання?

Лекція №2

Тема: ОСНОВНІ КОМПОНЕНТИ МОВИ VISUAL BASIC

Мета лекції: вивчення основних елементів і найпростіших конструкцій Visual Basic

Час: 2 години

Місце проведення: Аудиторія № Д, А

План

Вступ
2.1 Основні поняття при роботі з програмами
2.2 Алфавіт мови програмування VB. Типи даних
2.3 Використання імен у мові VB та правила їх утворення
2.4 Вирази та правила їх запису
2.5 Контрольні питання

м. Харків
2016р.

Вступ
2.1 Основні поняття при роботі з програмами

Перш ніж приступити до процесу програмування, треба визначити, які дії має виконувати програма та як описати їх за допомогою операторів і синтаксису мови Visual Basic (далі VB), мати основні поняття про дані, які необхідно обробляти під час виконання програми.

Під даними мають на увазі сукупність усієї інформації, яка обробляється в ході виконання програми.

Типом даних називається спосіб збереження і подання даних у комп'ютерній системі, що задає певний формат або розмір вмісту змінної.

У Visual Basic приблизно 200 вбудованих операторів і функцій. Кожен з них має чітку структуру (синтаксис), тобто правила граматики, пунктуації та орфографії. Для опису операторів застосовуються різні ключові слова.

Потрібно дотримуватися певних правил запису програм. У кожному рядку коду є оператор, який може мати додаткові параметри. Оператори можуть розташовуватися послідовно в одному рядку, тоді вони розділяються двокрапкою.

Логічний рядок можна розділяти на кілька фізичних. Розділь-ником рядків служить пропуск, який іде за символом підкреслення (_). Це дає можливість сформувати довгі рядки так, щоб вони повністю вміщувалися на сторінці екрану.

Рядок програми у Visual Basic може містити максимум 1023 символи і не більш як 10 роздільників.

У програмі можуть бути використані коментарі. Вони призначені для пояснення окремих фрагментів програми й ігноруються під час виконання програми. Для виділення початку коментарю можна використати або апостроф ('), або команду Rem, їхня дія однакова.

Rem – це оператор і тому він має знаходитися в окремому рядку. Апостроф можна ставити в будь-якому місці рядка, при цьому текст коментарю розташовують праворуч від цього символу.

Для багатьох процедур і функцій відображається підказка за синтаксисом (**ToolTip**).

2.2 Алфавіт мови програмування VB. Типи даних

Алфавіт мови програмування Visual Basic включає:

26 латинських літер: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z;

10 арабських цифр: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0;

26 символів: ! # $ % ^ & * ( ) + – / \ < > = ? @ _ : ; ' . , " пробіл.

Дані поділяються на три великі категорії: числові, рядкові та спеціальні.

Таблиця 2.1 – **Типи даних у Visual Basic**

Тип даних	Символ	Опис	Діапазон значень	Пам'ять
Byte		Двійкові дані	Від 0 до 255	1байт
Boolean		Логічний	True або False	2 байта
Integer	%	Цілі числа	Від –32768 до 32767	2 байта
Long	&	Цілі числа (довгі)	Від –2147483648 до +2147483647	4 байти
Single	!	Дійсні числа (одинарної точності)	Від –3.4E+38 до +3.4E+38	4 байти
Double	#	Дійсні числа (подвійної точності)	Від –1.79*D308 до +1.79*D308	8 байтів
String	$	Символьний	Від 0 до 65,4 тис. символів	10 байт+ довж ряд
Currency	@	Число з фіксованою десятковою крапкою	Від –922'337'203'685'477,5807 до 922337203685477,5807	8 байтів
Date		Дата	Від January 1,100 до December 31, 9999	8 байтів
Object		Об'єкт	Містить посилання на об'єкт	4 байти
Variant		Довільний тип	Будь-який з перерахованих	Від знач.

2.3 Використання імен у мові VB та правила їх утворення

Імена використовуються для позначення об'єктів у програмі.

Правила утворення імен:

- першим символом імені повинна бути літера;

- ім'я може включати тільки літери, цифри і символ підкреслення (_);

- в імені може бути не більше 40 символів;

- ключові слова або Reserved words (Зарезервовані слова) не можна використовувати як імена.

Бажано давати об’єктам імена, що відповідають контексту і несуть смислове навантаження.

Приклади.

Правильні імена	Неправильні імена
A2	23B (бо починається з цифри)
Color	CM*PER*INCH (бо є службові символи)
StartTime	File (бо ключове слово)

2.4 Вирази та правила їх запису

Вирази використовуються для виконання операцій над даними. В залежності від даних та операцій, які використовуються вирази можна поділити на арифметичні, логічні та символьні. Вираз можна визначити так:

операнд [знак операції] [операнд] ,

де в залежності від типу виразу використовуються відповідні операнди і знаки операцій.

Можна використовувати наступні знаки операцій:

+ – додавання (наприклад, 2.36+12.5);

- – віднімання (наприклад, 231-49);

* – множення (наприклад, 3*2);

^ – піднесення до ступеня (наприклад, $3\text{^}2 = 9$);

/ – ділення дійсних чисел (з плаваючою крапкою: 3/2=1.5);

\ – ділення цілих чисел (наприклад, 3\2=1);

Mod – вирахування остачі від ділення (7Mod4=3).

Пріоритет виконання операцій (в порядку зменшення пріоритету):

1. піднесення до ступеня;
2. множення та ділення дійсних чисел;
3. ділення цілих чисел;
4. вирахування остачі від ділення;
5. додавання та віднімання.

Обчислення у виразах здійснюються зліва направо. Пріоритет можна змінити за допомогою круглих (і тільки круглих) дужок.

Приклади.

14/5*2=5.6 – операції одного пріоритету виконуються зліва направо;

14\5*2=1 – множення має вищий пріоритет, а при діленні цілих чисел дробова частина числа відкидається;

$27\text{^}1/3 = 9$ – піднесення до ступеня має найвищий пріоритет;

$27\text{^}(1/3) = 3$ – дужки змінюють послідовність операцій.

2.5 Стандартні функції, які використовуються у VB

У VB є набір вбудованих (стандартних) функцій, що полегшує написання програм. Є математичні функції, для обробки рядків, для роботи з часом і датами, для фінансових розрахунків.

Основні математичні функції:

Abs(x)	модуль (абсолютна величина) числа x;
Atn(x)	арктангенс кута x;
Cos(x)	косинус кута x;
Fix(x)	ціле число, що дорівнює числу x без дробової частини;
Log(x)	повертає значення натурального логарифма;
Rnd(x)	генерує випадкове число між 0 та 1;
Sqr(x)	квадратний корінь з x;
Tan(x)	тангенс кута x;
Exp(x)	вираховує e^x.

Рядкові функції:

Len(x)	визначає довжину рядка;
Val(x)	перетворює рядок в число;
Str(x)	перетворює число в рядок;
Str()	створює рядок символів.

Функції для роботи з часом і датами:

Date	визначає і видає поточну дату;
Time	визначає і видає поточний час;
Day(x)	перетворює дійсне число в день місяця;
Month(x)	перетворює дійсне число в місяць року;
Year(x)	перетворює дійсне число в рік.

приклади запису математичних виразів за правилами VB. Математичний запис Запис за правилами Visual Basic

$y = \cos^2 x$	y=cos(x)^2
$d = 4e^{2x}$	d=4*exp(2*x)
$y = btgx^5$	y=b*tan(x^5)
$y_i = \sqrt{x_i}$	y(i)=sqr(x(i))
$y = \sqrt[5]{x}$	y=x^(1/5)
$y = \sin^3 x^2$	y=sin(x^2)^3
$y = \ln^5 x$	y=log(x)^5
$y = \lvert x^3 \rvert$	y=abs(x^3)
$t = \frac{a+b}{c}$	t=(a+b)/c

При запису математичних виразів за правилами Visual Basic необхідно враховувати такі функціональні конструкції:

Математичний запис	Запис за правилами Visual Basic
arcsin x	Atn(x/Sqr(1-x^2))
arccos x	1.57-Atn(x/Sqr(1-x^2))
arcctg x	Atn(x)+1.57
ctg x	1/Tan(x)
$\log_b a$	Log(a)/Log(b)
$\sqrt[n]{x}$	x^(1/n)

Записати самостійно наступні вирази за правилами Visual Basic:

$f = \frac{\cos x^6}{\sqrt[3]{x}} + |x| + tg^3 x$; $y = \sqrt[3]{\frac{x + tg^2 x}{a^2 + b^3}}$; $a = \frac{\cos^2 x^3}{e^2 + \ln^3 x}$;

$y = \sqrt{\cos^2 x + b}$; $y_i = \sqrt{tg(a^2 + x_i^2) + b^3}$; $t = \arccos^5\left(\frac{x}{7}\right)^3$;

$r = \frac{b^2 \sin^3 x}{|\ln x| + ax}$; $u = \frac{\sqrt[3]{\sin^2(x+a)}}{x+a}$; $y = \sqrt{|a^2 + b^2|}$.

Контрольні запитання

1. Як можна записати два оператори в одному рядку?
2. За допомогою яких символів можна продовжити запис довгого оператора на наступному рядку?
3. Який вигляд має оператор коментарю?

5. Які арифметичні операції можна виконувати з даними у Visual Basic?

6. Як можна змінити пріоритет операцій?

7. Які правила утворення імен змінних?

Лекція №3

Тема: ОСНОВНІ КОМПОНЕНТИ СЕРЕДОВИЩА ПРОЕКТУВАННЯ VISUAL BASIC. ЛІНІЙНІ ОБЧИСЛЮВАЛЬНІ ПРОЦЕСИ

Мета лекції: вивчити основні компоненти середовища проектування Visual Basic, методи програмування лінійних обчислювальних процесів

Час: 2 години

Місце проведення: Аудиторія № Д, А

План

Вступ

3.1 Елементи вікна середовища мови програмування VB

3.2 Послідовність завантаження Visual Basic.

3.3 Розгляд прикладу створення проекту в середовищі VB

3.4 Збереження проекту

3.5 Створення виконуваного exe-файлу

3.6 Контрольні запитання

м. Харків

2016р.

Вступ

Інтегроване середовище-оболонка – один із компонентів сучасної системи програмування Visual Basic. Воно включає вбудований редактор тексту, систему інформаційної контекстуальної допомоги, транслятор-компілятор, компонувальник і відлагоджувач програм, а також елементи призначеного для користувача інтерфейсу. Основні елементи інтегрованого середовища: рядок заголовка, рядок меню, панель інструментів, робоче поле, на якому знаходяться п’ять розкритих вікон (рис. 3.1).

3.1 Елементи вікна середовища мови програмування VB

Рядок заголовка (1) розміщується у верхній частині вікна проекту. В ньому знаходиться назва проекту і його екранної форми. У квадратних дужках вказується режим роботи – проектування або виконання. В правому кутку рядка заголовка є три стандартні для будь-якого додатку Windows кнопки управління вікном екрана: **Свернуть** (1.1), **Развернуть/Свернуть в окно** (1.2), **Закрыть** (1.3).

Рядок меню (2) знаходиться під рядком заголовка і містить кілька пунктів, кожний з яких має власне спадне меню, що відкривається клацанням лівою клавішею миші на цьому пункті. Спадне меню має список команд, які використовуються під час роботи з Visual Basic.

Рядок Панель інструментів (3) розташований під рядком меню, в ньому знаходяться команди із рядка меню , що найчастіше використовуються, у вигляді піктограм. Вони є швидким і зручним засобом використання команд.

У робочому полі середовища проектування Visual Basic на екрані розкрито, як правило, п’ять вікон :

1. Вікно екранної форми (4) – основна робоча область програміста у Visual Basic, яку проектувальник розробляє для користувача. У вікні знаходиться форма, зі стандартною сіткою, яка за замовчуванням називається **Form1** (Форма1). Розміри форми можна змінювати за допомогою миші, як і будь-яке вікно.

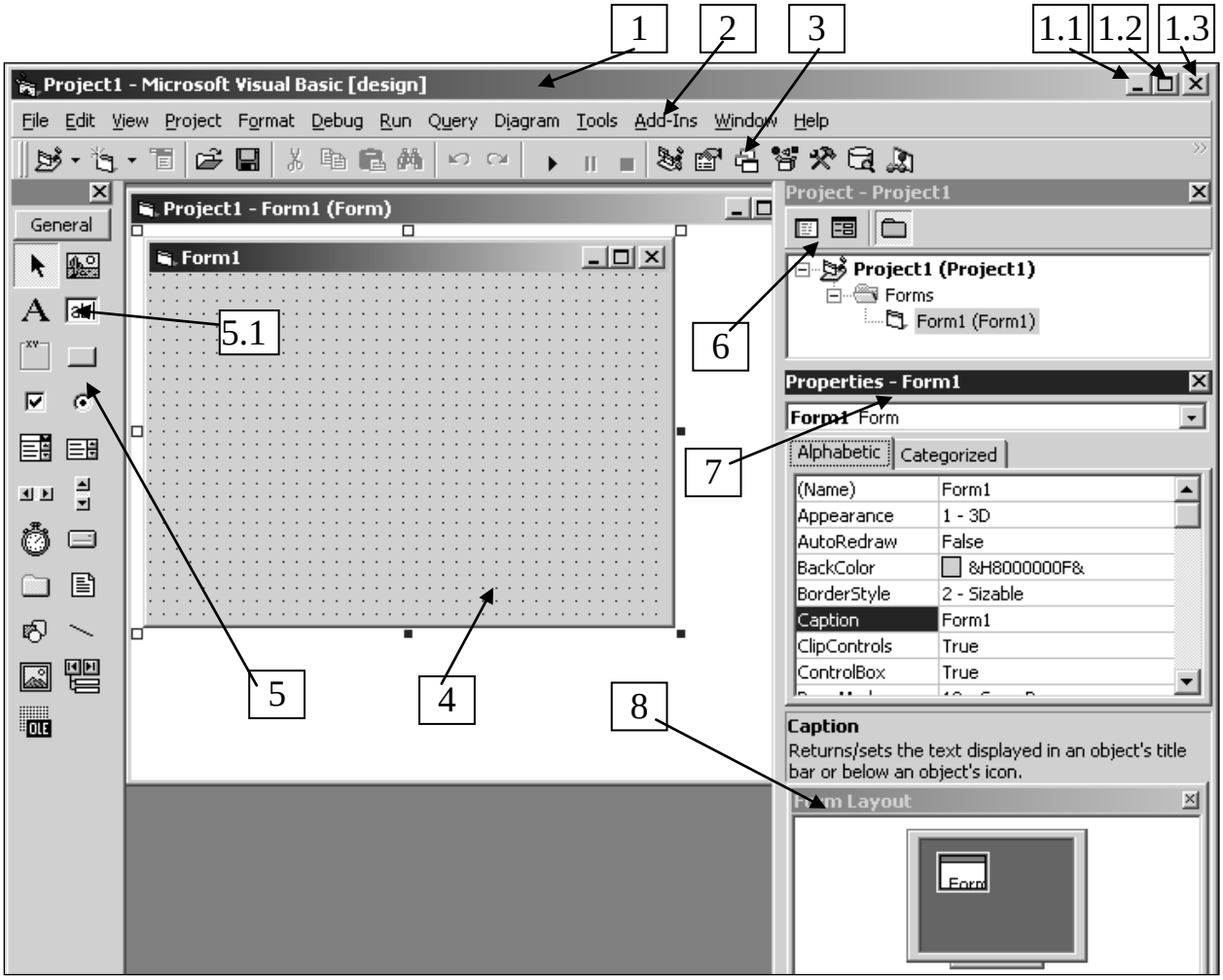


Рисунок 3.1 – Вигляд вікна середовища проектування Visual Basic після запуску

2. Вікно **Form Layout** (8) **–** призначене для встановлення місця розташування форми майбутньої програми, на робочому столі, після запуску програми для цього необхідно помістити зменшене зображення форми у потрібне місце вікна **Form Layout** (Схема Формы).

3. Вікно елементів управління **Toolbox** (General) (5) забезпечує проектувальника набором інструментів, необхідних для створення елементів вікна прикладної програми. Таб. 3.1.

Таблиця 3.1 Перелік інструментів для створення вікна програми

Назва англійською мовою	Назва російською мовою	Пік-тог-рама	Префікс (prefix)
Pointer	Указатель		
PictureBox	Графическое окно		pic
Label	Метка		lbl
TextBox	Текстовое поле		txt
Frame	Рамка		fra
Command Button	Командная кнопка		cmd
CheckBox	Контрольный индикатор		chk
Option Button	Кнопка-переключатель		opt
ComboBox	Комбинированное окно		cbo
ListBox	Окно списка		lst
HscrollBar	Горизонтальная полоса прокрутки		hsb
VscrollBar	Вертикальная полоса прокрутки		vsb
Timer	Таймер		tmr
Drive ListBox	Окно списка дисководов		drv
Directory ListBox	Окно списка каталогов		dir
File ListBox	Окно списка файлов		fil
Shape	Фигура		shp
Line	Линия		lin
Image	Изображение		img
Data	Данные		dat
OLE Container	OLE контейнер		ole

Елементи управління (5.1) – це візуалізований засіб для створення об'єктів екранної форми.

4. Вікно властивостей **Properties** (7) дає змогу змінювати властивості елементів інтерфейсу користувача. У рядку заголовка вікна вказується ім'я форми, до якої належить елемент управління. Із списку полів, що знаходиться під рядком заголовка, треба вибрати необхідний елемент управління. У списку розташованому нижче, перераховано властивості цього елемента (за алфавітом або за категоріями). Набір властивостей залежить від типу елементів управління. Список властивостей складається з двох стовпців: у лівому перераховано назви властивостей, а у правому – їх значення.

5. Вікно дослідника програми (**Project Explorer**) (6) призначене для відображення структури проекту, який розробляється, і перемикання його окремих компонентів. Структура проекту відображається у вигляді «дерева», що схоже на структуру папок, з яких складається проект Visual Basic.

Вікно редактора коду **Code** сховане за вікном екранної форми. Це вікно, в якому мовою Visual Basic записується програмний код, що складається з операторів мови, констант та опису даних.

Якщо вікна екранної форми та редактора коду закриті, їх можна розкрити за допомогою кнопок, розташованих у вікні **Дослідника програми**, вибором команди **Object** (Объект) чи **Code** (Код) з меню **View** (Вид).

Код додатка Visual Basic поділяється на дрібніші блоки, які називаються процедурами. Для створення процедури обробки події потрібно зі списку об'єктів вікна редактора коду вибрати елемент, з яким пов'язана процедура, що розробляється, а зі списку процедур – елемент із ім'ям події для вибраного об'єкта. Між операторами **Sub** та **End Sub** треба записати програмний код. З кожною подією пов'язана процедура її обробки – фрагмент коду, який виконується після здійснення певної події.

Ім'я процедури обробки події для елемента управління складається з істинного його імені (заданого у властивості **Name** (Имя)), символу підкреслення (_) та імені події-процедури.

Структура Windows-додатка:

- кожна екранна форма містить власний код form module (тобто модуль форми), який керує і реагує на елементи цієї форми; ім'я файлу форми має розширення **.frm**;

- у стандартному модулі зберігаються процедури загального призначення, які можуть бути викликані з модуля форми процедурами подій. Ім'я стандартного файлу має розширення **.bas**;

- проект може складатися з однієї або кількох екранних форм і одного чи кількох програмних модулів. Проект зберігається в окремому файлі, що має розширення **.vbp**. У проекті ведеться облік усієї колекції файлів, які утворюють Windows-додаток.

Створення Windows-додатка у середовищі Visual Basic складається з таких 4 етапів:

1) розробка ескізу екранної форми і сценарію взаємодії користувача з комп'ютером; розташування на формі елементів управління і задання властивостей для них;

2) розробка програмного коду; введення програмного коду в комп'ютер;

3) запуск програми на виконання (відлагодження, якщо є помилки); збереження розробленого додатка на диску у вигляді файла форми з розширенням **.frm**, та файла проекту з розширенням **.vbp** (бажано з одним і тим же ім'ям, щоб не плутатись);

4) перетворення проекту в exe-файл.

3.2 Послідовність завантаження Visual Basic

Для цього необхідно виконати команду **Пуск ⇒ Программы ⇒ Microsoft Basic Studio 6.0 ⇒ Visual Basic 6.0**. Або клацнути по піктограмі Visual Basic на Робочому столі чи на панелі задач в полі швидкого запуску. На екрані відкривається вікно **New Project** (**Новый проект**) рисунок 3.2. У вікні **New Project** (**Новый проект**) (вкладка **New** (**Новый**)) вибрати стандартний додаток VB (**Standard EXE**) і натиснути кнопку **Ok** (**Открыть**). На екрані відкривається вікно **Project1** (**Проект1**) з формою **Form1** (**Форма1**) рисунок 3.1. Вигляд вікна середовища проектування Visual Basic після запуску.

Рисунок 3.2 – вікно New Project (Новый проект)

3.3 Розгляд прикладу створення проекту у середовищі VB

Розглянемо на прикладі створення програмного проекту та схему алгоритма для розв’язання задачі обчислення середньої швидкості автомобіля для таких початкових даних: $L = 200$ км; $T = 4$ год.

Розробимо схем алгоритму рішення лінійної задачі отримаємо результат рисунок 3.3.

2. Розробимо ескіз форми майбутнього проекту (рис. 3.4): для цього нам необхідно проаналізувати, які об'єкти нам необхідно створити на формі. Це чотири об’єкти класу **Label** (Метка), три об’єкти класу **TextBox** (Текстовое поле) і один об’єкт класу **CommandButton** (Командная кнопка).

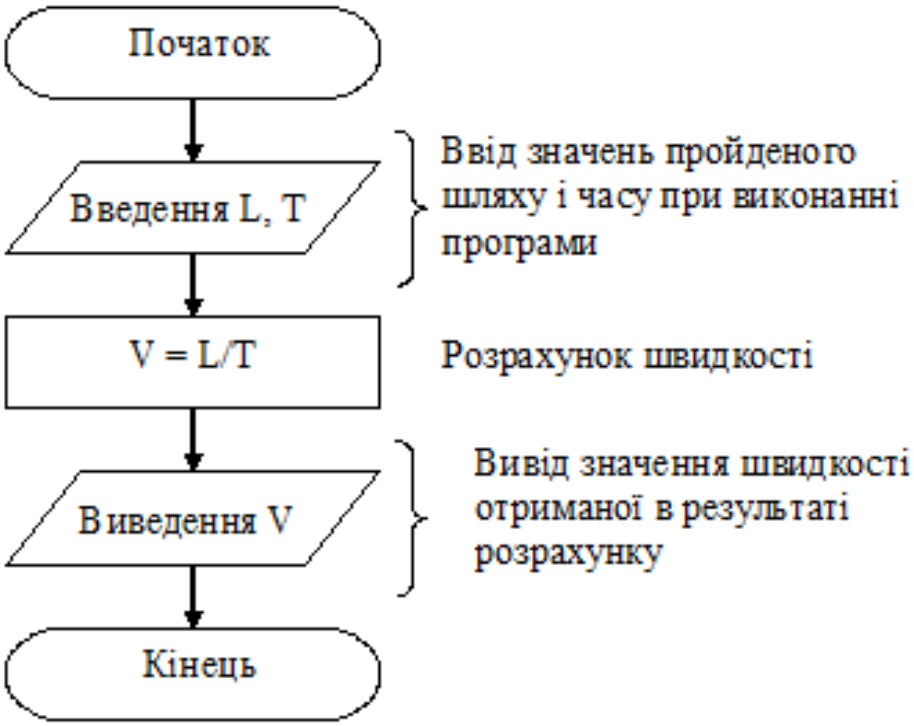


Рисунок 3.3 – Схема алгоритма обчислення середньої швидкості автомобіля

3. Встановити ім’я форми, напис в рядку заголовка:

- задати у вікні властивостей (**Properties**) ім’я форми (властивість **Name** (Имя)) **Швидкість**;

- у текстовому вікні поруч з назвою властивості **Caption** написати **Обчислення швидкості,** це присвоєння імені вікну створюваного додатку.

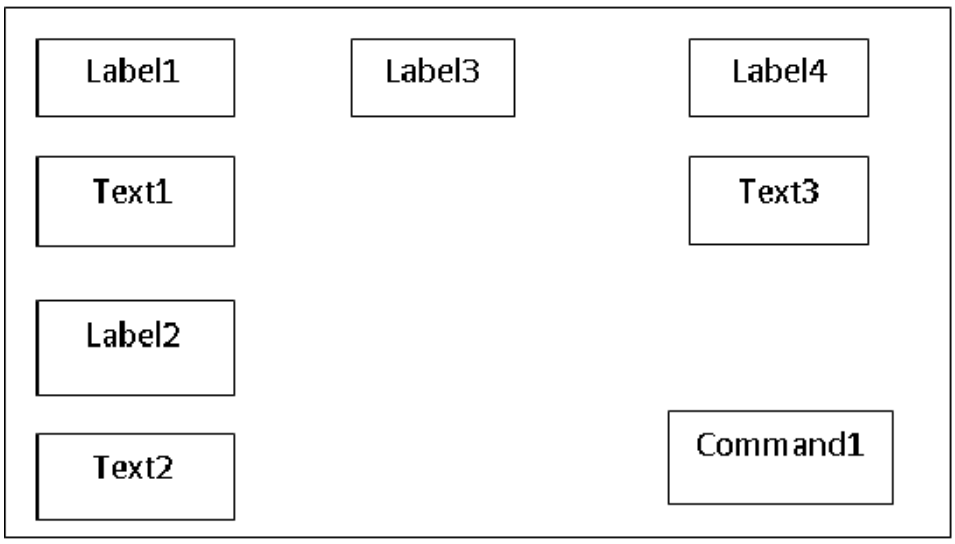


Рисунок 3.4 – Ескіз форми проекту

4. Установити об’єкт екранної форми **Label** (Метка):

- у вікні інструментів **Toolbox** (General) клацнути піктограму **Label** – це кнопка з великою літерою  **,** таблиця 3.1 (кнопка буде візуально утоплена);

- помістити покажчик миші (він набуде вигляду +) в те місце, де буде знаходитися лівий верхній кут майбутнього об’єкта, з натиснутою лівою клавішею перетягнути покажчик у місце, де буде знаходитися правий нижній кут майбутнього об’єкта. Цей процес буде супроводжуватися розтяганням пунктирної рамки – межі створюваного об’єкта. Розміри і положення прямокутника, що з’явився, можна коректувати;

- відпустити кнопку миші і на формі з’явиться об’єкт із написом **Label1** (Метка1.

5. Установити значення властивостей об’єкта **Label1** (Метка1).

Вибрати об’єкт **Label1**. По периметру об’єкта з’являться розмірні маркери (вісім чорних квадратиків). Це означає, що даний об’єкт є активним і у вікні **Properties** (Свойства) відображається список властивостей цього об’єкта (мітки).

Ім’я (**Name** (Имя)) об’єкта залишити дане системою за замовчуванням, тобто **Label1** (Метка1).

Дати назву мітці – у рядку властивості **Caption** написати **Відстань.**

6. Установити об’єкт екранної форми **TextBox** (Текстовое поле) другим способом. Для цього:

- клацнути двічі піктограму **TextBox** (другу зверху в правому стовпчику піктограм вікна **Toolbox** – вікна з написом **General**) – кнопку з написом  ;

- перемістити об’єкт **Text1**, що з’явився в центрі форми під об’єкт **Label1** рис. 3.3;

- довільно вирівняти розміри об’єкта.

7. Встановити значення властивостей об’єкта **Text1.**

Властивості **Name** (Имя) залишити ім’я дане системою за замовчуванням, тобто **Text1** (Текст1 – у русифікованій версії).

У текстового поля НЕМАЄ властивості **Caption**, у вікно текстового поля можна вводити великий обсяг інформації, а можна про-

сто числа. Використовуючи клавіші **Del** чи **Backspace**, очистити рядок праворуч від властивості **Text**.

8. Установити усі об’єкти та їх властивості згідно табл. 3.2.

Таблиця 3.2 – **Значення властивостей об’єктів управління для завдання 3**

Об’єкт управління	Ім’я	Властивість	Значення
Label2	Label2	Caption	Час
Label3	Label3	Caption	V=L/T
Label4	Label4	Caption	Швидкість
TextBox1	Text1	Text	Очистити від тексту
TextBox2	Text2	Text	Очистити від тексту
TextBox3	Text3	Text	Очистити від тексту
CommandButton1	Command1	Caption	Старт

Після встановлення усіх об’єктів на формі та їх властивостей, ми отримаємо наступний вигляд форми Рисунок 3.5.

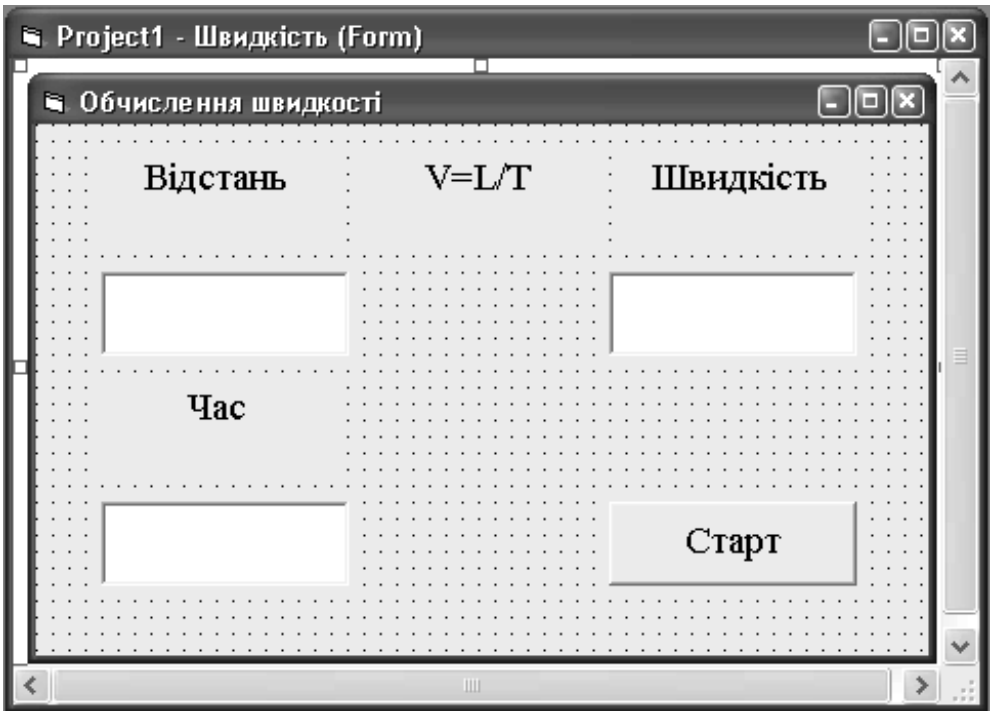


Рисунок 3.5 – Вид вікна форми після виконання пункту 8

9. Підготувати текст програми (програмний код).

Для написання програмного коду і прив’язки його до певної події необхідно розкрити вікно програмного коду (**View→Code**). Програмний код майже завжди прив’язується до якої-небудь події.

Двічі клацнути об’єкт, для якого треба написати код, у даному

разі це командна кнопка. У вікні коду з'явиться так звана заготовка, що стосується об'єкта **CommandButton**, це відображено в першому текстовому вікні. Поруч записана подія, до якого відноситься даний код, – **Click.**

Набрати з клавіатури текст програмного коду між першим і останнім рядками заготовки. Одержати наступний вигляд програмного коду:

```
Private Sub Command1_Click()
L=Text1.Text
T=Text2.Text
V=L/T
Text3.Text=V
End Sub
```

9. Відлагодити проект і усунути помилки.

Запустити програму на виконання клавішею **F5** або натиснути кнопку **Start** (Начать) на панелі інструментів.

У текстове вікно **Text1** (Текст1) ввести 200, а в текстове вікно **Text2** (Текст2) ввести 4. Натиснути кнопку **Старт**. Програма почне виконуватись. В текстовому полі **Text3** виведеться результат. Виправити помилки (якщо вони є).

3.4 Збереження проекту

Для цього необхідно:

- Виконати команду **File→Save Form As...**,

У діалоговому вікні **Save Form As** в текстовому полі **Папка** встановити поточним диск D:.Слайд 18

- - На диску D: створити папку (ім'я папки – назва групи, наприклад, А-11), використовуючи кнопку **Создание новой папки**, зробити створену папку робочою (двічі клацнути по піктограмі створеної папки).

- У діалоговому вікні **Save Form As** в текстовому полі **Имя файла** ввести: **Швидкість**, якщо воно не задано.

- Переконатися, що у текстовому полі **Тип файла** записано **Form Files (*.frm)**.

- Натиснути кнопку **Сохранить**.

- Виконати команду **File→Save Project As…**.

- У діалоговому вікні **Save Project As** в текстовому полі **Имя файла** ввести **Швидкість**.

- Переконатися, що у текстовому полі **Тип файла** записано **Project Files (*.vbp)**.

- Натиснути кнопку **Сохранить**. Слайд 19.

3.5 Створення виконуваного exe-файлу

- клацнути команду **File** (Файл) головного меню;

- у наступному меню клацнути команду **Make Швидкість.exe** (Создать Швидкість.exe);

- у вікні, що з'явилося, перевірити ім'я і клацнути **Ok**.

Створений Файл **Швидкість.exe** дозволяє подвійним клацанням на ньому запускати створений додаток без завантаження Visual Basic.

1. Перелічити основні елементи інтегрованого середовища Visual Basic.
2. Які і скільки вікон розкрито після запуску середовища проектування Visual Basic?
3. Як складається ім'я процедури обробки події для елемента управління?
4. Які етапи необхідно виконати для створення проекту?
5. Чим відрізняється ім'я форми від імені проекту?
6. Яка ознака активного об'єкта на формі?
7. Як можна перемістити об'єкт? Як змінити розміри об'єкта?
8. Які властивості об'єктів типу **Label** були використані в роботі?
9. Які властивості об'єктів типу **TextBox** були використані в роботі?
10. Які властивості об'єкта **CommandButton** були використані в роботі?
11. Яка головна характеристика об'єкта **CommandButton**?
12. Як зберегти проект?

Лекція №4

Тема: РОЗГАЛУЖЕНІ ОБЧИСЛЮВАЛЬНІ ПРОЦЕСИ

Мета лекції: вивчити формат запису операторів управління та методів програмування розгалужених обчислювальних процесів

Час: 2 години

Місце проведення: Аудиторія № Д, А

План

Вступ
4.1 Розгалужені обчислювальні процеси VB
4.2 Багаторядкова (блочна) форма функції умови.
4.3 Оператори If … Then … ElseIf
4.4 Формат оператора Select Case
4.5 Використання функції для вводу даних InputBox
4.6 Контрольні запитання

м. Харків
2016р.

4.1 Розгалужені обчислювальні процеси VB

Розгалужений обчислювальний процес (РОП) (або обчислювальний процес, що розгалужується, або обчислювальний процес із розгалуженням) передбачає вибір одного з кількох можливих варіантів обчислень в залежності від результату перевірки умови, яка має вигляд виразу логічного типу. Внаслідок перевірки цієї умови можливі два варіанти: **True** (Істина) – умова виконується; **False** (Хибність) – умова не виконується.

Складні порівняння здійснюються за допомогою логічних операторів **And**, **Or**, **Not**. Ці оператори дають змогу об'єднати дві або більше умови в одну.

Для організації вибору застосовується умовний **оператор If**. Існують дві форми цього оператора одно- та багаторядкова (блочна). Однорядкова має такий формат запису:

If умова Then оператор_1 [Else оператор_2]

У квадратних дужках – конструкції, які можуть бути відсутні. Якщо ключове слово **Else** відсутнє і умовний вираз має істинне значення, то робиться перехід до виконання операторів після слова Then. В разі хибного значення умовного виразу виконання складного оператора відразу припиняється і починають виконуватися оператори, записані в наступних рядках програми. Такий умовний перехід називається одинарним.

Приклад, використання однорядкового оператора:

Умова 1.

$z=\begin{cases} ax, & \text{якщо } 0 \le x < 0{,}5 \\ (2+x)b, & \text{якщо } x >= 0{,}5 \end{cases}$	A	B	X
	4,01	1,05	Змінна

Для того, виконати розрахунок необхідно ввести значення змінних a, b, x. Значення змінним a, b присвоїмо використовуючи оператор присвоєння. Значення змінній x будемо присвоювати при виконанні програми, використовуючи оператор InputBox. Для виводу результатів обчислень (значення функції z) використаємо оператор List.box. Тому для організації взаємодії користувача з нашою програмою необхідно створити на формі два елементи рис. 4.1.

Створення форми

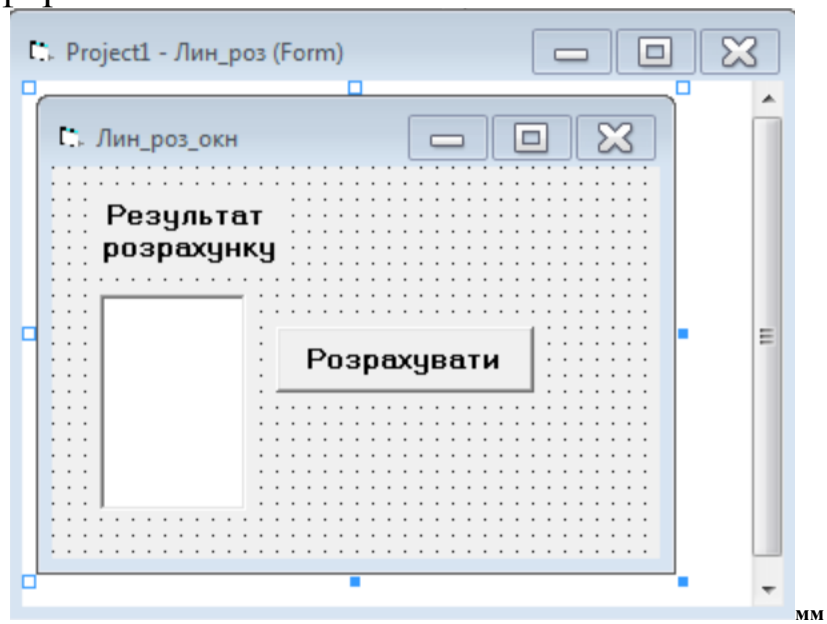


Рисунок 4.1 – Форма для рішення задачі умова1

Текст програмного коду

```
Private Sub Command1_Click()
a=4.01
b=1.05
x= InputBox("Введіть X", "Значення X", 0.5)
If x>=0 and x<0,5 Then z=a*x Else z=(2+x)*b
List1.AddItem(z)
End Sub
```

4.2 Багаторядкова (блочна) форма функції умови

Формат запису

If умова Then
оператори_1
Else
оператори_2
End If

Коли умовний оператор містить ключове слово **Else** і умовний вираз має істинне значення, здійснюється перехід до виконання операторів після слова **Then**. Якщо ж умовний вираз має хибне значення, то робиться перехід до виконання операторів після слова **Else**. **End If** означає, що виконання умовного оператора припиня-

ється і починають виконуватися оператори програми, записані в наступних рядках.

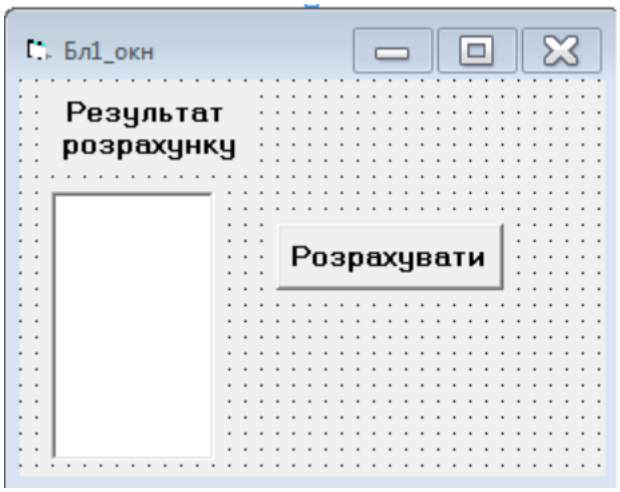
Розглянемо, приклад складання програмного коду блочного оператора.

Для розробки програмного коду використаємо умову, що використовували для лінійного оператора if.

A	B	X
4,01	1,05	Змінна

$$z = \begin{cases} ax, & \text{якщо } 0 \le x < 0,5 \\ (2+x)b, & \text{якщо } x >= 0,5 \end{cases}$$

Екранна форма буде мати такий же вигляд, як і в першому прикладі.




```
Private Sub Command1_Click()
a = 4.01
b = 1.05
x = InputBox("Введіть X", "Значення X", 0.3)
If x >= 0 And x < 0.5 Then
z = a * x
Else
z = (2 + x) * b
End If
List1.AddItem (z)
End Sub
```

4.3 Оператори If … Then … ElseIf

Оператори **If … Then … ElseIf** дають можливість розглянути, крім головної умови, ще й додаткові. Додаючи нові оператори **ElseIf,** можна розглянути будь яку кількість умов. Цей формат можна назвати складна блочна форма

Формат оператора:
If головний_умовний_вираз **Then**
послідовність_операторів_A
ElseIf умовний_вираз_1 **Then**
Послідовність_операторів_1
ElseIf умовний_вираз_2 **Then**
послідовність_операторів_2
ElseIf умовний_вираз_3 **Then**
послідовність_операторів_3
…
ElseIf умовний_вираз_N **Then**
послідовність_операторів_N
[**Else** послідовність_операторів_B]
End If

Спочатку обчислюється значення головної умови в операторі **If**. Якщо воно дорівнює **True**, то виконується оператор (чи оператори) після оператора **Then**. Далі програма переходить до виконання оператора, записаного безпосередньо після оператора **End If**.

Якщо перша умова має значення **False**, то програма відразу переходить до виконання першого оператора **ElseIf**, щоб перевірити значення його умови. Якщо значення цієї умови дорівнює **True**, то будуть виконані оператори записані за цим оператором, після чого програма переходить до виконання оператора, записаного безпосередньо після оператора **End If**. В іншому разі ця послідовність дій повторюється до наступного оператора **ElseIf**, і так триває доти, доки не будуть розглянуті всі оператори.

Якщо всі умови мають значення **False** (неправда), то програма переходить до оператора **Else** і виконуються команди, розташовані між **Else** та **End If**. Присутність оператора **Else** не обов'язкова. Розглянемо на прикладі використання оператора **If … Then … ElseI**f.

Приклад 2.

Умова:

$$z=\begin{cases} ax & ,якщо \quad 0\le x<0,5 \\ (2+x)b & ,якщо \quad 0,5\le x<1 \\ (x+c)^2+b & ,якщо \quad 1\le x\le 2 \end{cases}$$

A	B	C
4,01	1,05	-2,08

Вікно форми залишимо таке, як було у попередньому прикладі в ньому додамо вікно List.Box, щоб контролювати введені значення змінної x. Ми не будемо змінювати організацію присвоєння значень змінним і організацію виводу результатів обчислень. Введемо надписи формі, щоб позначити, де будуть розташовані вхідні параметри і де вихідні. Вікно форми буде мати вигляд рисунок 4.2.

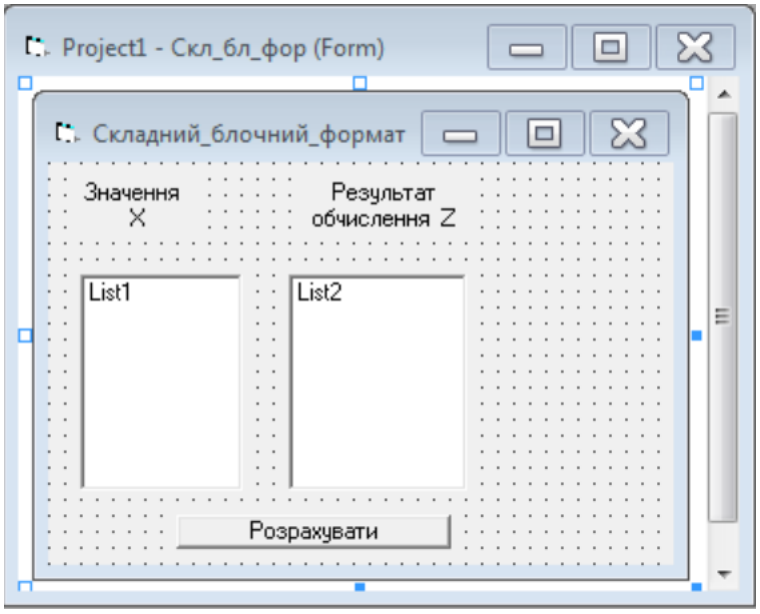


Рисунок 4.2 – вікно форми для прикладу 2

Текст програмного коду

```
Private Sub Command1_Click()
a = 4.01
b = 1.05
c = -2.08
x = InputBox("Введіть X", "Ввід значення X", 0.5)
List1.AddItem (x)
If x >= 0 and x < 0.5 Then
Z = a * x
ElseIf x >= 0.5 and x < 1 Then
Z = (2 + x) * b
ElseIf x >= 1 and x <= 2 Then
```

```
Z = (x + c) ^ 2 + b
Else
Z = "X не входить в діапазон"
End If
List2.AddItem (Z)
End Sub
```

Використання вкладених операторів IF в блочній формі

Якщо потрібно перевірити яку-небудь додаткову умову в разі, якщо головна умова є істинною, то використовуються вкладені оператори If.

Формат вкладеного оператора:

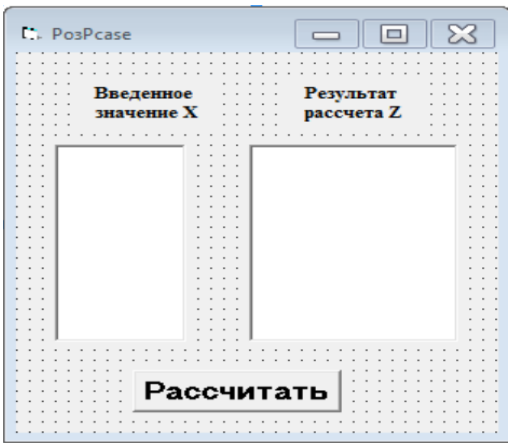
If умова_1 **Then**
If умова_2 **Then**
оператори_A
Else
оператори_B
End If
End If

Розглянемо приклад. Умова – необхідно обчислити значення змінної z.

$$z = \begin{cases} ax & \text{,якщо } 0 \le x < 0{,}5 \\ (2+x)b & \text{,якщо } 0{,}5 \le x < 1 \\ (x+c)^2 + b & \text{,якщо } 1 \le x \le 2 \end{cases}$$

A	B	C
4,01	1,05	-2,08

Форма додатку буде мати вигляд такий, як у попередньому прикладі



Код програми

```
Private Sub Command1_Click()
a = 4.01: b = 1.05: c = -2.08
x = InputBox("Введіть X ", " Ввід значення X", 0.3)
List2.AddItem (x)
If x >= 0 And x <= 2 Then
 If x < 0.5 Then
 z = a * x
 Print z
 End If
If x >= 0.5 And x < 1 Then
z = (2 + x) * b
Print z
End If
If x >= 1 And x <= 2 Then
z = (x + c) ^ 2 + b
End If
Else
z = " X не входить в діапазон "
End If
List1.AddItem (z)
End Sub
```

4.4 Формат оператора Select Case

У Visual Basic є зручний засіб, за допомогою якого розв’язують задачі вибору одного варіанту з їх безлічі. Цим засобом є **оператор Select Case**.

Формат оператор такий:

Select Case вираз
Case діапазон_значень_1
послідовність_операторів_1
Case діапазон_значень_2
послідовність_операторів_2
…
Case Else

послідовність_операторів_N

End Select

Виразом можуть бути значення однієї змінної або їх комбінація.

Діапазон_значень_1, діапазон_значень_2, … можуть набувати одного із значень виразу, можуть бути переліком кількох значень, розділених комою, або діапазоном значень.

Логіка роботи оператора **Select Case** така. Спершу обчислюється значення виразу. Знайдене значення виразу порівнюється із значеннями, записаними в діапазонах значень після ключового слова **Case**. Виконуватися буде та послідовність операторів, для яких значення виразу збігається зі значеннями, записаними в діапазоні значень.

Діапазон значень можна задавати трьома способами:

- перерахуванням значень через кому;
- за допомогою ключового слова **To**, ліворуч від якого записується найменше, а праворуч – найбільше значення;
- за допомогою ключового слова **Is**, праворуч від якого записуються знак порівняння і деяке значення.

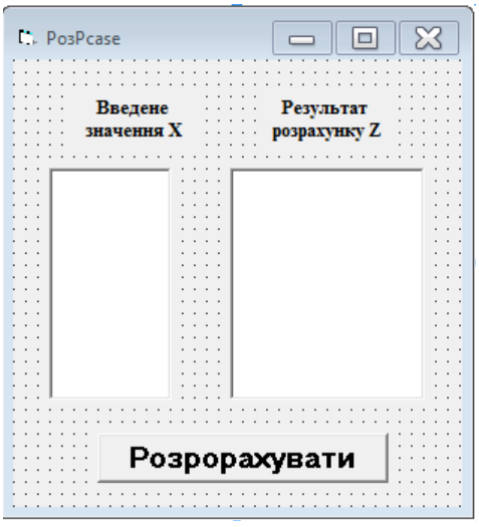
Якщо умові **Select Case** відповідає кілька блоків **Case**, виконується перший з них.

Приклад використання оператора **Select Case**.

$$z=\begin{cases} ax & \text{, якщо } 0 \le x < 0{,}5 \\ (2+x)b & \text{, якщо } 0{,}5 \le x < 1 \\ (x+c)^2+b & \text{, якщо } 1 \le x \le 2 \end{cases}$$

A	B	C
4,01	1,05	-2,08

Вікно форми буде мати вигляд:



Рішення реалізуємо за допомогою оператора **Select Case.**

```
Private Sub Command1_Click()
a = 4.01: b = 1.05: c = -2.08
x = InputBox("Введіть X", "Увід значення X", 0.3)
List1.AddItem (x)
Select Case x
Case Is < 0
z = "X в недіапазонні"
Case Is > 2
z = " X в недіапазонні"
Case 0 To 0.5
z = a * x
Case Is < 1
z = (2 + x) * b
Case Is <= 2
z = (x + c) ^ 2 + b
End Select
List2.AddItem (z)
Print z End Sub
```

Зовнішній вигляд форми

4.5 Використання функції для вводу даних InputBox

За допомогою **функції InputBox** створюються вікна введення даних підчас виконання програмного коду. Вони використовуються, коли користувач повинен відповідати на певні типи питань, а також для присвоєння значень змінним з якими програма буде виконувати обчислення. Вікна введення завжди надають користувачеві місце для відповіді на питання.

Синтаксис функції **InputBox**:

InputBox ("Запрошення"[, "Заголовок"][, ПочЗначення]),

де Запрошення – це будь-який текст, який повинен знаходитися в вікні введення. Його призначення – служити підказкою користувачеві, яку інформацію він повинен ввести в спеціальне поле введення, що знаходиться в цьому вікні; Заголовок – заголовок в рядку заголовка текстового вікна; ПочЗначення – рядкове значення за умовчанням, яке VB відображає для відповіді за умовчанням, і користу-

вач може прийняти цю відповідь або змінити її.

Контрольні запитання

1. Який обчислювальний процес є розгалуженим?
2. Що таке розгалуження обчислень?
3. Чи є обов’язковою частина **Else** в операторі розгалуження (умовному операторі)?
4. Для чого використовувалася функція **Формат**?
5. Навести формат оператора **If**.
6. За допомогою яких логічних операторів здійснюються складні порівняння?
7. Навести формат оператора **Case**.
8. Для чого використовується функція **InputBox**?
9. Для чого використовується функція **MsgBox**?

Лекція №5

Тема: ЦИКЛІЧНІ ОБЧИСЛЮВАЛЬНІ ПРОЦЕСИ

Мета лекції: вивчити призначення, формат запису, способи використання операторів організації циклів, методи програмування циклічних обчислювальних процесів.

Час: 2 години

Місце проведення: Аудиторія № Д, А

План

Вступ

1 Циклічні обчислювальні процеси оператор циклу **For … Next**

2 Ітераційні цикли Формат запису циклів з перед умовою та післяумовою

3 Використання ключових слів While? Until

4 Розгляд методів створення ітераційних циклів

5 Метод організації негайного виходу із циклу

6 Контрольні запитання

м. Харків
2016р.

1 Циклічні обчислювальні процеси, оператор циклу For … Next

Циклічний обчислювальний процес – це такий процес обробки інформації, для якого характерним є багаторазове виконання одного або кількох ділянок алгоритму. Багато разів повторювані процеси називають циклами або повтореннями.

Розрізняють два види циклів: з відомим і невідомим числом повторень.

Для програмування повторень із заздалегідь відомим їх числом застосовується оператор циклу **For … Next**, який називається циклом з лічильником. Такі цикли ще називають арифметичні цикли.

Формат багаторядкового оператора **For … Next:**

For Ім’я=Значення_1 **To** Значення_2 [**Step** Значення_3]

Оператори, що повторюються (Тіло цикла)

Next [Ім’я]

Тут Ім’я – це ім’я змінної, яку називають лічильником (індексом циклу), або управляючою змінною;

Значення_1 – початкове значення лічильника;

Значення_2 – його кінцеве значення;

Значення_3 – величина, на яку змінюється значення лічильника за одне повторення. Вона називається кроком циклу.

Оператори, що повторюються (тіло цикла) – це та частина програми, яка має повторитися один або кілька разів. Розташована, між операторами **For** та **Next.**

Конструкція [**Step** Значення_3] може бути відсутня. При цьому за замовчуванням вважається, що лічильник змінює своє значення на одиницю (крок циклу дорівнює 1).

Ім’я лічильника після ключового слова **Next** також може бути відсутнім.

Цикл завершується тоді, коли значення лічильника стає більшим, ніж його кінцеве значення. Коли вказується від’ємний крок, цикл завершується, якщо значення лічильника стає меншим, ніж його кінцеве значення.

Кількість повторень циклу обчислюється по формулі:

$$\mathbf{K_{повт} = \frac{X_к - X_п}{D_X} + 1,}$$

де **Хк** – кінцеве значення управляючої змінної, яке вона повинна досягти;

Хп – початкове значення управляючої змінної, яке їй присвоюється на початку циклу;

Dx – крок, або величина на яку змінюється управляюча змінна, після виконання однієї дії з операторами тіла циклу.

Розглянемо використання оператора **For … Next** на прикладі розробки проекту для розрахунку функції F умова приведена в таблиці 5.1.

Таблиця 5.1 - **Дані для розрахунку функції F**

Функція F	a	b	xn	xk	dx
$F = b\sqrt{tg(x + a^3) + b}$	0,10	0,01	0,54	1,40	0,20

Складемо схему алгоритму у відповідності з умовою Рисунок 5.1

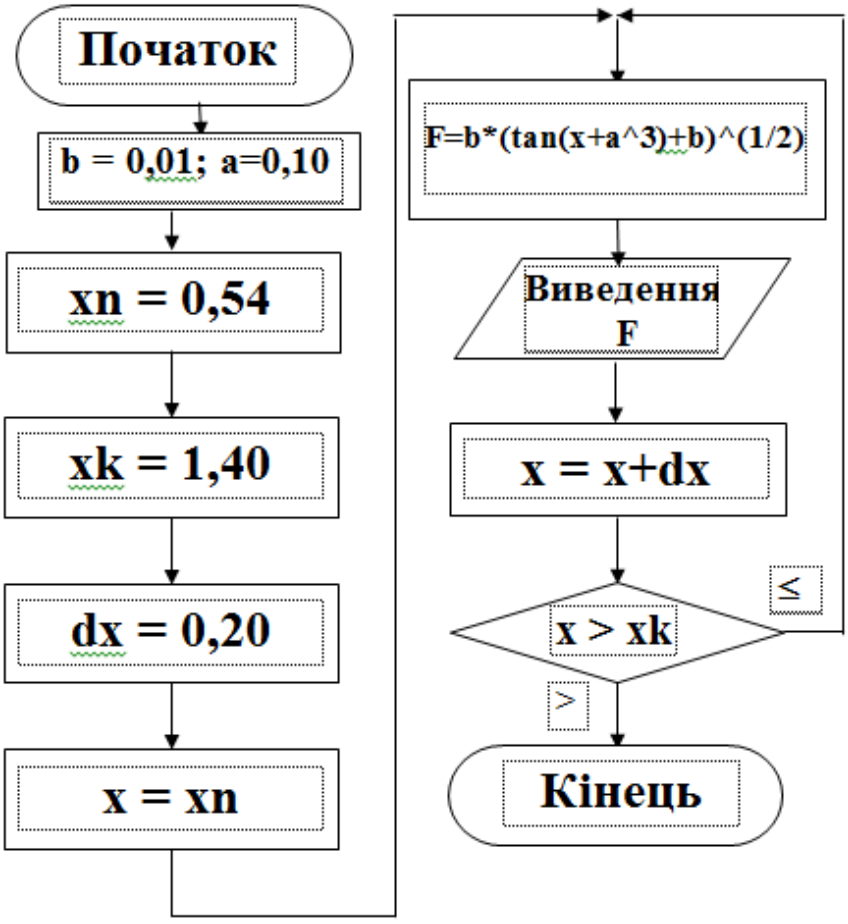


Рисунок 5.1 – Алгоритм розрахунку функції F

Тепер необхідно скласти схему розташування елементів на формі майбутнього проекту розрахунку функції Рисунок 5.2.

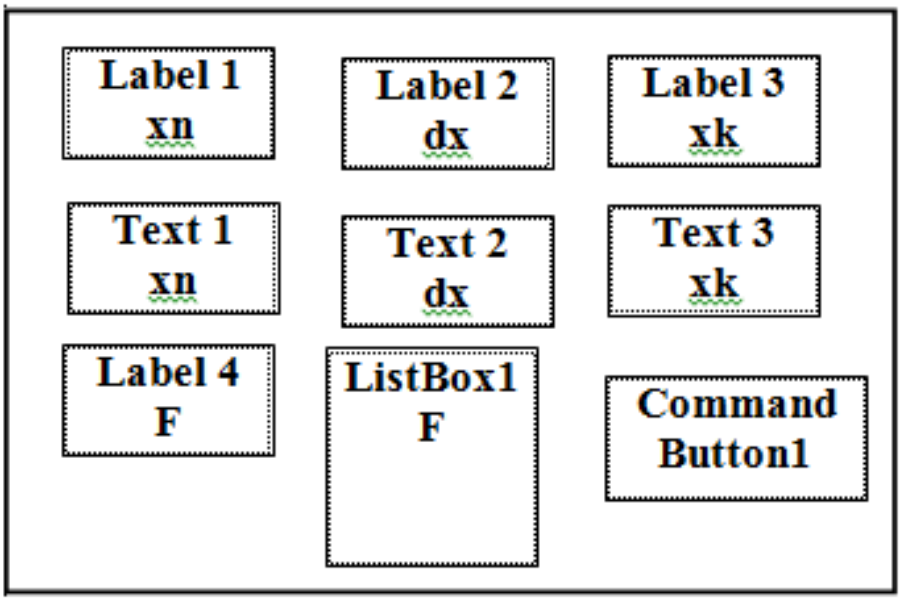


Рисунок 5.2 – Схема розташування елементів управління на формі

Після цього ми переходимо до складання програмного коду. Програмний код починаємо складати з команд виконання дії запуску програмного коду на виконання, а в середині послідовність виконання програмного коду у відповідності з складеним алгоритмом. Програмний код буде мати наступний текст.

Private Sub CmdF_Click() b=0.01 a=0.1 xn=Txtxn.Text xk=Txtxk.Text dx=Txtdx.Text For x=xn To xk Step dx f= b*sqr(tan(x+a^3)+b) LstF.AddItem f Next End Sub	Private Sub Command1_Click() b = 0.01 a = 0.1 xn = Text1.Text xk = Text3.Text dx = Text2.Text For x = xn To xk Step dx f = b * sqr(Tan(x + a ^ 3) + b) List1.AddItem f Next End Sub
В програмному коді управляючим елементам присвоєні імена розробником	В програмному коді імена управляючим елементів використані по замовчуванню (ті, які використовуються в середовищі мови VB автоматично)

Після цього ми запускаємо середовище Visual Basic заповнюємо форму елементами управління, створюємо програмний код, перевіряємо роботу програми та якщо необхідно виправляємо помилки і зберігаємо програмний продукт, або передаємо його на використання.

Реалізована форма на VB, буде мати наступний вигляд, рисунок 5.3.

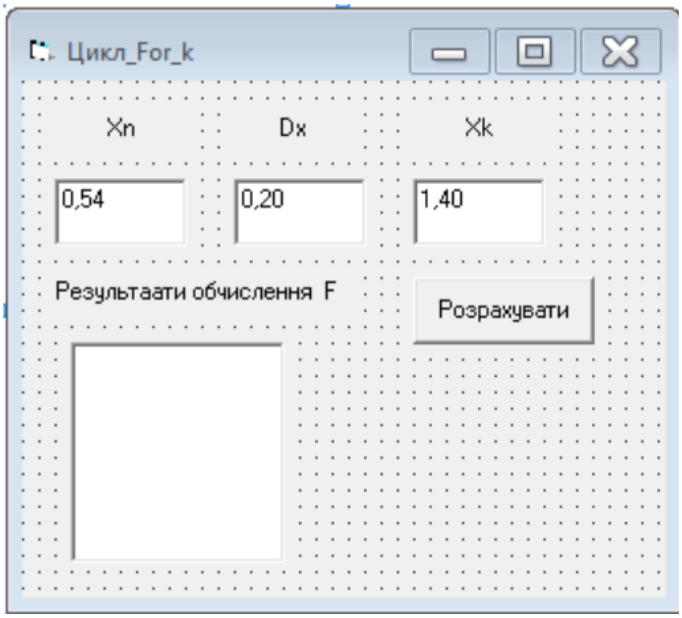


Рисунок 5.3 - Форма проект реалізована для циклу FOR

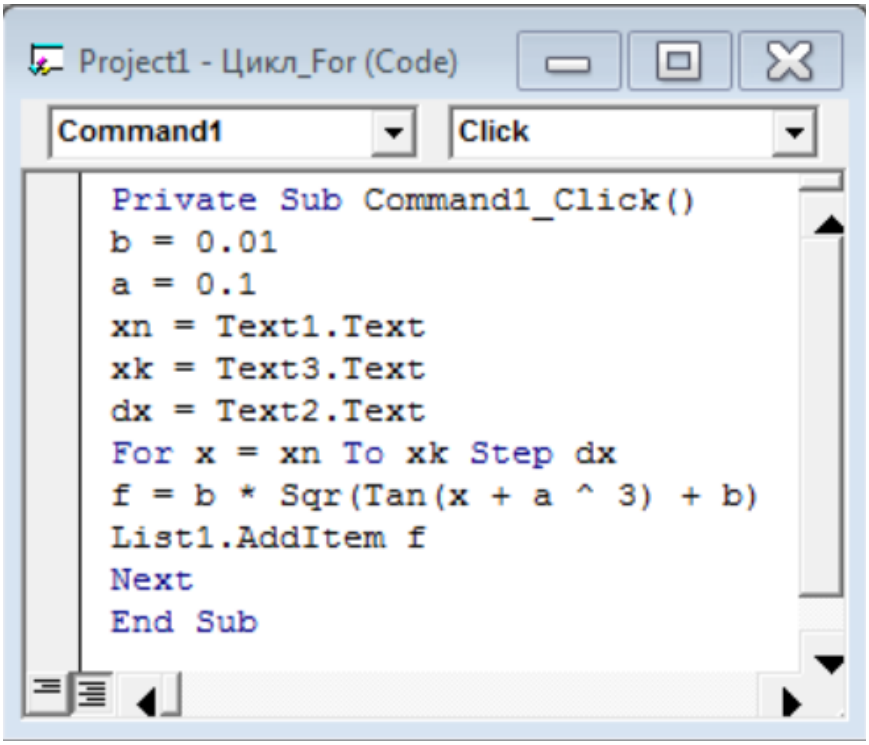


Рисунок 5.4 – Програмний код

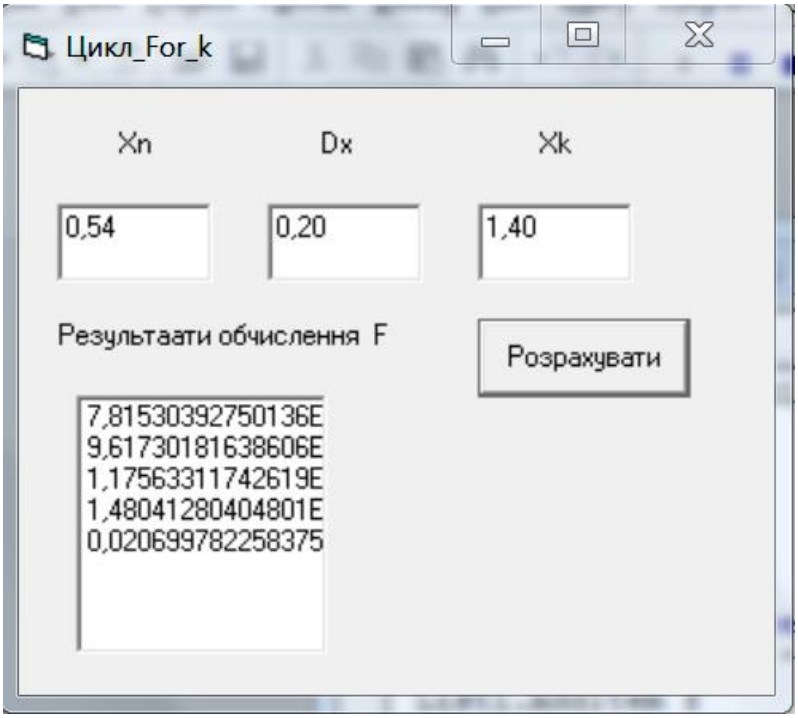


Рисунок 5.5 – Результат обчислень

2 Ітераційні цикли Формат запису циклів з передумовою та післяумовою

Для програмування циклічних процесів обробки інформації із заздалегідь невідомим числом повторень (їх ще називають ітераційними) використовуються різні форми оператору циклу з умовою.

Ітераційні цикли мають два типи - цикли з **передумовою** і цикли з **післяумовою.**

Цикл з перед умовою – це багаторядковий оператор, перший рядок якого починається з ключового слова Do (виконати), а останній з ключового слова Loop (цикл).

Формати ітераційних циклів мають два різновиди.

Перший з передумовою:

Do Умова

Оператори, що повторюються

Loop

Другий формат циклу з післяумовою:

Do

Оператори, що повторюються

Loop Умова

Ці дві форми циклу з умовою різняться розташуванням умови, яка буває двох типів:

- з ключовим словом While (воно називається умовою продовження циклу);

- з ключовим словом Until (воно називається умовою завершення циклу).

Після ключового слова While (або Until) записується умовний вираз.

3 Використання ключових слів While? Until

У циклі з передумовою в якому використовується ключове слово While спочатку перевіряється умова, якщо умова істинна, то виконуються оператори циклу і знову здійснюється повернення до оператора Do While та перевіряється умова. Якщо ж умова хибна, то всі оператори, записані в циклі, не виконуються і відбувається вихід з циклу. Цикл із такою конструкцією може виконуватися будь-яку кількість раз, поки значення умови є істиною. Оператори циклу не виконуються жодного разу, якщо при першій перевірки умови вона виявляється хибною.

При використанні ключового слова Until, оператори, що повторюються виконуються, якщо значення умовного виразу є хибним. В іншому випадку цикл завершується.

У циклах з післяумовою оператори циклу обов’язково виконуються один раз незалежно від того яке ключове слово використано While чи Until і яке значення має умова істина чи хибність.

При використанні ітераційних циклів необхідно організовувати зміну значень управляючих змінних в середині циклу щоб цикл мав логічне завершення, а не зациклювався (повторювався до безконечності, тобто його не можливо буде зупинити).

Для того, щоб завершити цикл негайно, не продовжуючи подальших операцій, використовується оператор **Exit For**, який може знаходитися в середині оператора **If … Then** або **Select Case**. Ці оператори повинні знаходитись в середині тіла циклу і в них повинна бути записана умова в якому випадку (при якій умові) необхідно зупинити виконання циклу.

4 Приклади використання ключових слів While або Until в програмних кодах

Приклади використання ітераційних циклів розглянемо на прикладі, який ми використовували для знаходження функції **F**, коли розглядали використання оператора циклу **For…next**. Замість цього оператора ми розглянемо приклади використання операторів операційних циклів.

Приклад. Перший запис розрахунку функції **For** з передумовою і використанні ключових слів **While** та **Until.** Порівняйте програмні коди.

Програмний код арифметичного циклу з FOR	Програмний код з передумов Ключове слово While
Private Sub Command1_Click() b = 0.01 a = 0.1 xn = Text1.Text xk = Text3.Text dx = Text2.Text For x = xn To xk Step dx f = b * sqr(Tan(x + a ^ 3) + b) Next End Sub	Private Sub Command1_Click() b = 0.01 a = 0.1 xn = Text1.Text: xk = Text3.Text dx = Text2.Text: x=xn Do While xk >=x f = b * sqr(Tan(x + a ^ 3) + b) List1.AddItem f X=x+dx Loop End Sub

Програмний код з FOR	Програмний код з передумов Ключове слово Until
Private Sub Command1_Click() b = 0.01 a = 0.1 xn = Text1.Text xk = Text3.Text dx = Text2.Text For x = xn To xk Step dx f = b*sqr(Tan(x + a ^ 3) + b) Next End Sub	Private Sub Command1_Click() b = 0.01 a = 0.1 xn = Text1.Text xk = Text3.Text dx = Text2.Text x=xn Do Until xk <=x f = b*sqr(Tan(x + a ^ 3) + b) List1.AddItem f X=x+dx Loop End Sub

Програмний код з післяумовою. Ключове слово While.	Програмний код з післяумовою. Ключове слово Until.	
Private Sub Command1_Click() b = 0.01 a = 0.1 xn = Text1.Text xk = Text3.Text dx = Text2.Text x=xn Do f = b*sqr(Tan(x + a ^ 3) + b) List1.AddItem f X=x+dx Loop While xk <=x End Sub	Private Sub Command1_Click() b = 0.01:a = 0.1 xn = Text1.Text xk = Text3.Text dx = Text2.Text x=xn Do f = b*sqr(Tan(x + a ^ 3) + b) List1.AddItem f X=x+dx Loop Until xk <=x End Sub	

Контрольні запитання

1. Які два види циклів розрізняють?
2. Яке призначення операторів циклу?
3. Для чого в додатку використовувалося вікно **ListBox**?
4. Що таке лічильник циклу?
5. Яке значення має лічильник циклу після його завершення?
6. Для чого використовується ключове слово **Step**?
7. В якому разі відсутнє ключове слово **Step**?
8. Скільки разів виконується рядок For…To…[step] при виконанні циклу?
9. Для чого призначений оператор Next?
10. Записати два різновиди формату ітераційного циклу
11. Що необхідно враховувати при програмуванні ітераційних циклів, щоб вони "не зациклювались"?
12. Який оператор використовується для негайного завершення циклу?

Лекція №6

Тема: ОБЧИСЛЮВАЛЬНІ ПРОЦЕСИ З МАСИВАМИ ДАНИХ

Мета лекції – вивчити поняття масиву, елементи масиву, організацію обробки та зберігання масивів даних.

Час: 2 години

Місце проведення: Аудиторія № Д, А

План

Вступ

1 Поняття масиву, елементи масиву

2 Типи даних. Оголошення масиву

3 Приклад створення програми при роботі з одновимірним масивом

4 Приклад створення програми при роботі з масивами, яка складається з окремих модулів

5 Особливості роботи з двовимірними масивами даних

м. Харків
2016р.

1 Поняття масиву, елементи масиву

Масив – упорядкований набір однотипних елементів, що має одне ім'я. За визначенням усі елементи масиву мають однаковий тип. Однак бувають винятки: коли типом даних масиву є тип **Variant**, тоді елементи масиву можуть містити дані різних типів (об'єкти, рядки, числа).

Кожний елемент масиву має свій **порядковий номер (індекс). Кількість індексів визначає розмірність масиву**. Масиви можуть бути **одномірними (вектори), двовимірними (матриці), тривимірними** і т. д.

Нижні та верхні межі зміни індексу масиву визначають кількість його елементів і називаються граничною парою і визначають **розмір масиву**. Елементи масиву безперервно розташовуються між заданими межами.

Для зберігання масиву виділяється неперервна область оперативної пам'яті, в якій розташовуються всі його елементи. *Ємність пам'яті, яка виділяється для збереження масиву, залежить від кількості елементів масиву і їх типу.*

Згідно з визначенням усі елементи масиву мають однакове ім'я, але відрізняються місцем розташування в масиві і значенням, тому їх імена теж відрізняються номером, який вказує місце розташування елемента в масиві його називають індексом. *Звідси і формат(правило) запису імен елементів масиву:* **Ім'я масиву(індекс).** Індекс може бути змінною, або числом, які можуть приймати значення цілого додатного числа (нумерація починається з нуля).

Для багатовимірних масивів: **Ім'я_масиву(список індексів)**

Список індексів – це індекси, між якими ставиться кома. В разі двовимірного масиву перший індекс це номер рядка таблиці, а другий – номер її стовпця. У тривимірному масиві додається третій індекс – номер самої таблиці.

Наприклад, масив середня добова температура тижня. Ім'я масиву СДТТ тоді імена елементів масиву їх сім (одновимірний масив) можна записати як: СДТТ(0), СДТТ(1), СДТТ(2), СДТТ(3), СДТТ(4), СДТТ(5), СДТТ(6). З елементом масиву можна виконувати всі дії, як з окремою змінною - присвоювати значення, виконувати арифметичні та логічні операції і т.п.

Таким чином масив характеризується: *типом даних, іменем, розміром, розмірністю, місцем знаходження та датою створення чи редагування.*

2 Типи даних. Оголошення масиву

У операторі **Dim** після ключового слова **As** можна оголосити тип змінної:

Dim Ім'яЗмінної **As** НайменуванняТипу

У цьому операторі НайменуванняТипу може бути надано одним з ключових слів: Byte, Integer, Long, Single, Double, Currency, Date, String, Boolean, Object, Variant (див. лабораторну роботу №1). Наприклад:

Dim A **As** Byte

Dim Filename As String

Масив фіксованої довжини оголошується так само, як і змінна, але вживаються круглі дужки після імені змінної, в яких вказуються діапазони зміни значень індексів. Наприклад:

Dim A(1 to 10) **As** Integer

Dim МатрицяВ(1 to 3, 1 to 5)

Найчастіше замість діапазону записується просто число (**Dim** A(9) **As** Integer). Номери індексів починаються з нуля. Оператор **Option Base** дає змогу задати індексацію масиву з 1. Він має знаходитись в секції **General Declarations** (Общие описания загальне описання) контейнера (форми, модуля).

У VB розрізняють статичні та динамічні масиви.

Межі статичного масиву встановлюються на етапі розробки програми і можуть змінюватися тільки в новій версії програми.

Динамічні масиви змінюють свої межі в ході виконання програми. За їх допомогою можна динамічно задавати розмір масиву відповідно до конкретних умов.

3 Приклад створення програми при роботі з одновимірним масивом

Виконати приклад. Створити проект для обчислення значень елементів масиву Y_i за відомими значеннями елементів масиву X_i $(i = 1, 2, ..., 6)$ за формулою:

$$Y_i = 2 \cdot X_i + A,$$

де $A = 2{,}7$, а елементи масиву X любі дійсні числа. Значення елементам масиву присвоює користувач проекту підчас виконання програми.

Для розробки проекту необхідно виконати наступні дії:

1. Ознайомитись із схемою алгоритму рис. 6.1 і замалювати її в зошит. Осмислити, які типи даних будуть використовуватись, як будуть відображатись вихідні дані та результат на формі та як організувати їх обробку в програмі.

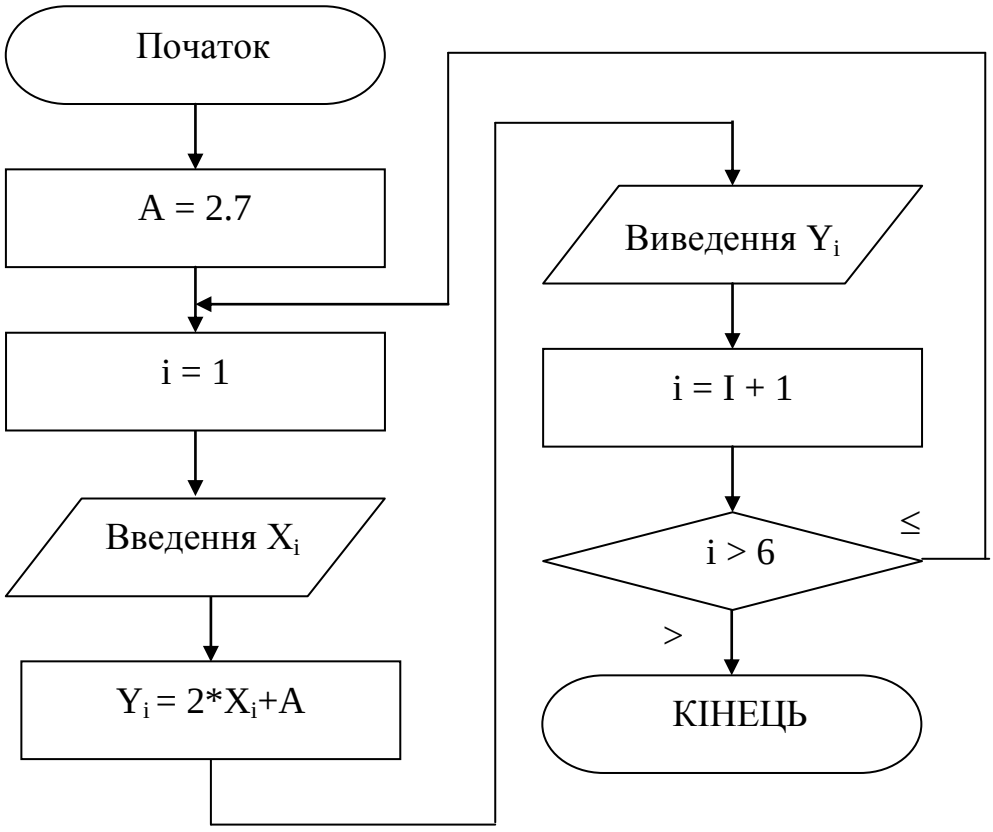


Рисунок 6.1 – Схема алгоритму розрахунку елементів масиву X_i

2. Розробити ескіз форми майбутнього проекту (рис. 6.2)
3. Запустити Visual Basic.
4. Заповнити властивості форми:
Font (Шрифт): Arial, Начертание: курсив, Размер: 11
Name (Имя) Масиви
Caption Обчислення елементів масиву
BackColor &H00C0C0FF& (Розовый)

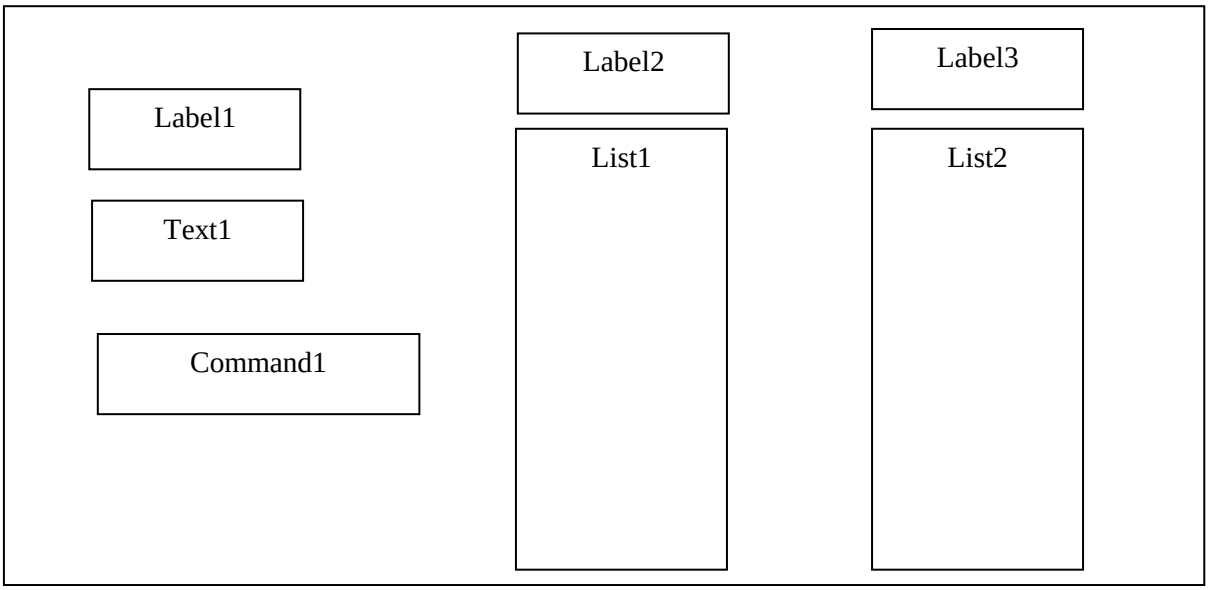


Рисунок 6.2 – Ескіз форми для завдання 1

5. Розташувати на формі об’єкти управління (згідно з ескізом) і у вікні **Properties** задати значення властивостей об’єктів управління (табл. 6.1).

Таблиця 6.1 – **Значення властивостей об’єктів управління для завдання 1**

Об’єкт управління	Ім’я	Властивість	Значення
Label1	LblA	Caption	Значення A
Label2	LblX	Caption	Масив Xi
Label3	LblY	Caption	Масив Yi
TextBox1	TxtA	Text	2,7
ListBox1	LstX	List	Очистити від тексту
ListBox2	LstY	List	Очистити від тексту
CommandButton1	CmdYi	Caption	Старт

6. Двічі клацнути на кнопці **Старт**, у вікні редактора коду

(**Code**) ввести текст програмного коду. Одержати такий вигляд програмного коду:

```
Option Explicit
Option Base 1
Dim x(6) As Single
Dim y(6) As Single
Dim A As Single
Dim i As Byte
Private Sub CmdYi_Click()
Print "Обчислити y(i)=2*x(i)+A"
A=TxtA.Text
   For i=1 To 6
       x(i)=InputBox("Ввести елемент масиву x" _
       +Str(i),"вводимо_елементи_масиву_x")
       LstX.AddItem(x(i))
       y(i)=2*x(i)+A
      LstY.AddItem(y(i))
   Next i
End Sub
```

4 Приклад створення програми при роботі з масивами, яка складається з окремих модулів

Створити проект та схему алгоритму для обчислення значень елементів масиву Y_i за відомими значеннями X_i, при цьому ввід елементів масиву X буде виконано окремим модулем, розрахунок елементів масиву Y окремим і закінчення роботи з програмою окремим модулем. Обчислити значення елементів масиву за формулою:

$$y_i = \begin{cases} b + \sin^3 x_i, & \text{якщо} \quad x_i < 1 \\ b \cdot x_i^2, & \text{якщо} \quad x_i \geq 1 \end{cases},$$

де $b = 5{,}3;\ x_i = \{0{,}2;\ 2{,}1; 1{,}3; 0{,}6; 3{,}4; 2{,}3; 0{,}2; 1\}$.

1 Розробити схему алгоритму

2 Розробити ескіз форми майбутнього проекту (рис. 6.3).

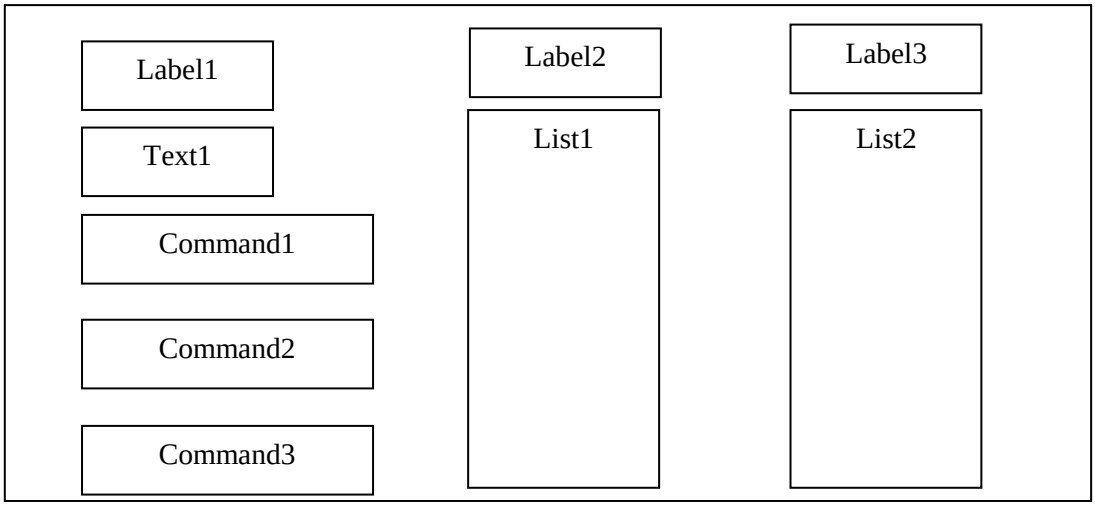


Рисунок 6.3 – Ескіз форми для завдання 2

3 Виконати команду **File→New Project** (Файл→Новый проект).

4 Розташувати на формі об’єкти управління (згідно з ескізом) і у вікні **Properties** задати значення властивостей об’єктів управління (табл. 6.2).

Таблиця 6.2 – **Значення властивостей об’єктів управління для прикладу 2**

Об’єкт управління	Ім’я	Властивість	Значення
Label1	LblB	Caption	Значення B
Label2	LblX	Caption	Масив Xi
Label3	LblY	Caption	Масив Yi
TextBox1	TxtB	Text	5,3
ListBox1	LstX	List	Очистити від тексту
ListBox2	LstY	List	Очистити від тексту
CommandButton1	CmdВвод	Caption	Ввести Xi
		ToolTipText	Натиснути, щоб почати вводити вихідні дані
CommandButton2	CmdСтарт	Caption	Старт
CommandButton3	CmdEnd	Caption	Закрити

Властивість **ToolTipText** містить текст, що з’являється як підказка під час виконання додатка.

5 Двічі клацнути на кнопці **Старт**, у вікні редактора коду (**Code**) ввести текст програмного коду. Одержати такий вигляд програмного коду:

```
Option Explicit
Option Base 1
```

```
Dim x(8) As Single
Dim y(8) As Single
Dim i As Byte
Dim b As Single
Private Sub CmdСтарт_Click()
For i=1 To 8
  If x(i)<1 Then
   y(i)=b+sin(x(i))^3
       Else
   y(i)=b*x(i)^2
  End If
  LstY.AddItem(y(i))
Next i
End Sub
```

6 Двічі клацнути на кнопці **Ввести Xi**, у вікні редактора коду (**Code**) ввести між заготовками код програми наступного вигляду:

```
Private Sub CmdВвод_Click()
b=InputBox("Ввести b", "введення b")
LblB.Caption=("b="+Str(b))
For i=1 To 8
        x(i)=InputBox("Ввести елемент масиву x" _
        +Str(i),"вводимо_елементи_масиву_x")
        LstX.AddItem(x(i))
Next i
```

7 Двічі клацнути на кнопці **Закрити**, у вікні редактора коду (**Code**) ввести між заготовками код програми наступного вигляду:

```
End
```

8 Доопрацювати форму.

Для чого:

Відкрити форму і виконати форматування командних кнопок:

- утримуючи клавішу **Shift** виділити всі три командні кнопки;
- виконати команду **Format→Make Same Size→Both** (Форматировать→Размер→Оба) та **Format→Align→Centers** (Форматировать→Выровнять→По центру);
- у вікні властивостей (**Properties**) змінити значення властивості:

а) Style на **1-Graphical** (Графический);

б) BackColor на **&H00FFC0C0&** (Сиреневый).

в) Font (Шрифт) на **Times New Roman**, Начертание: **жирный**, Размер: **14**.

Виконати форматування міток, змінюючи за своїм вибором властивості **Alignment**, **BackColor**, **ForeColor**. Вирівняти їх розміри та розташування.

9 Відкомпілювати програму і перевірити її виконання. Введені числа, повний код проекту та відповіді записати в зошит.

10 Зберегти проект з іменем **Комбі** у папці, ім'я якої – Ваше прізвище на робочому столі у папці "Мои документы".

5 Особливості роботи з двовимірними масивами даних

Для створення масиву використаємо генератор випадкових чисел від 0 до 10 розмірністю 3 на 4.

Розробимо алгоритм створення масиву рис. 6.4 замалювати його в зошит і розібратись як він працює.

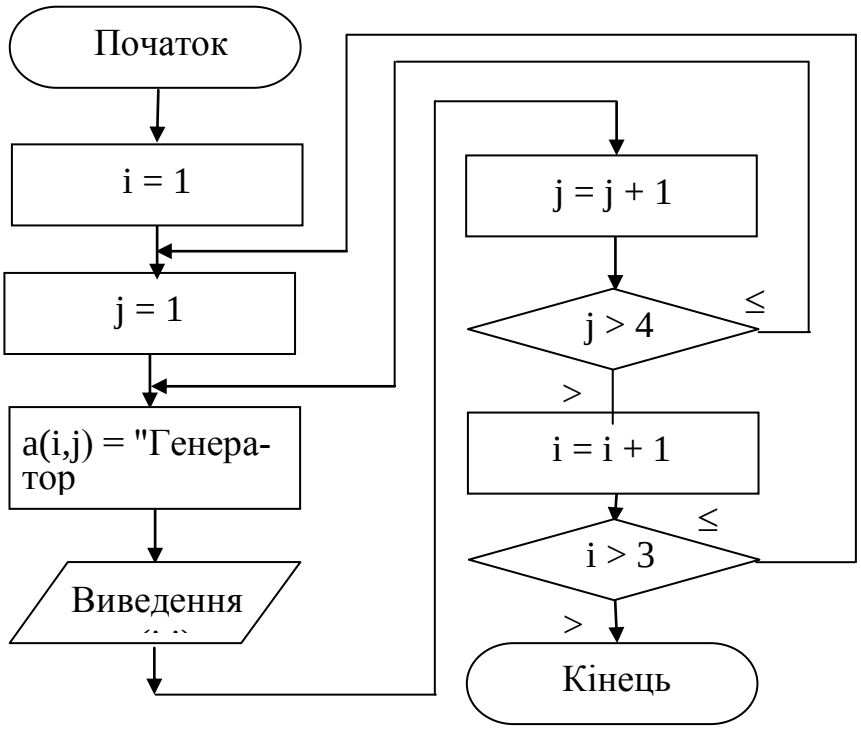


Рисунок – 6.4 Схема алгоритму присвоєння значень елементам масиву та вивід цих значень на форму

1. Виконати команду **File→New Project** (Файл→Новый проект).

2. Внести у вікно форми командну кнопку **CommandButton1** з назвою **Ввод_даних**, присвоїти властивості **Name** (Имя) **CmdВвод**.

3. Двічі клацнути на кнопці **Ввод_даних**, у вікні редактора коду (**Code**) ввести текст програмного коду.

```
Option Explicit
Option Base 1
Dim a(3,4) As Single
Dim i As Byte
Dim j As Byte
Private Sub CmdВвод_Click()
Cls
For i=1 To 3
For j=1 To 4
a(i,j)=Int(Rnd(10)*10)
Print a(i,j);
Next j
Print
Next i
End Sub
```

Оператор **Cls** очищує вікно екранної форми від інформації, яка на неї виведена та розміщує курсор у лівому верхньому куті форми.

Самостійно:

Відкомпілювати програму і перевірити її виконання.

Зберегти проект з іменем **Двовимірний_масив** у папці, ім’я якої – Ваше прізвище.

6 Контрольні запитання

1. Що таке масив?
2. Як оголосити тип змінної?
3. Яким оператором описують масив?
4. Для чого потрібна властивість **ToolTipText**?
5. Що таке розмірність масиву?
6. Які оператори використовують для введення масиву?
7. Для чого потрібний оператор **Option Base 1**?
8. В якому операторі вказують кількість елементів масиву?

ЛІТЕРАТУРА

1. Глушаков С.В., Мельников В.В., Сурядный А.С. Программирование в среде Windows: Учебный курс. – Харьков: Фолио; М.: АСТ, 2000. – 487 с.

2. Інформатика: Комп'ютерна техніка. Комп'ютерні технології. Підручник / За ред. О.І.Пушкаря – Київ: Академія, 2002. – 704 с.

3. Малышев С.М. Самоучитель VBA. – СПб.: Наука и техника, 2001. –496 с.

4. Гарнаев А. Самоучитель VBA. – СПб.: BHV. – Санкт-Петербург, 2002. – 464 с.

5. Браун С. Visual Basic 6. Учебный курс. – СПб.: Питер, 2007. – 574 с.

6. Котікова М.В., Скрипіна І.В., Кудін А.І., Шевченко В.О. Методичні вказівки до лабораторних робіт з дисциплін: «Інформатика», «Комп'ютерна техніка та програмування» для студентів за напрямами підготовки «Автомобільний транспорт», «Транспортні технології», «Будівництво» «Екологія, охорона навколишнього середовища та збалансоване природокористування» (розділ: «Інтегроване середовище розробки Visual Basic») – Харків: ХНАДУ, 2010. – 86 с.