В.М. Бутенко, В.С. Меркулов,
О.В. Казанко, О.В. Чаленко

ОСНОВИ ПРОГРАМУВАННЯ МОВАМИ ВИСОКОГО РІВНЯ

НАВЧАЛЬНИЙ ПОСІБНИК

Харків 2009

УКРАЇНСЬКА ДЕРЖАВНА АКАДЕМІЯ
ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

В.М. Бутенко, В.С. Меркулов,
О.В. Казанко, О.В. Чаленко

ОСНОВИ ПРОГРАМУВАННЯ МОВАМИ ВИСОКОГО РІВНЯ

НАВЧАЛЬНИЙ ПОСІБНИК

Рекомендовано Міністерством освіти і науки України
як навчальний посібник для студентів
вищих навчальних закладів

Харків 2009

ББК 65.9(2)21

О
УДК 6684.3.06

Основи програмування мовами високого рівня: Навч. посібник / В.М.Бутенко, В.С.Меркулов, О.В.Казанко, О.В.Чаленко. – Харків: УкрДАЗТ, 2009. – 206 с.

ISBN 978-966-2033-10-6

У навчальний посібник включені матеріали лекцій, лабораторно-практичних занять, тести, результати досліджень, виконані співробітниками, кафедри ОТіСУ Української державної академії залізничного транспорту при викладанні дисциплін „Обчислювальна техніка, програмування, моделювання систем”, „Комп’ютерна техніка та програмування”.

Викладені конструкції на мовах програмування QBasic та Visual Basic. Особлива увага приділяється питанням використання методики викладання матеріалу для студентів некомп’ютерних напрямків підготовки.

Посібник призначений для студентів та слухачів вищих навчальних закладів транспорту.

Іл. 20, табл. 19, бібліогр.: 24 назв.

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів (№ 1.4/18-Г-65 від 10 січня 2009 року)

Рецензенти:

доктор техн. наук, професор Л.В.Дербунович (Національний технічний університет „ХПІ”)

доктор техн. наук, професор Г.Ф.Кривуля (Харківський національний університет радіоелектроніки)

В.М. Бутенко, В.С. Меркулов,
О.В. Казанко, О.В. Чаленко

ОСНОВИ ПРОГРАМУВАННЯ МОВАМИ ВИСОКОГО РІВНЯ

НАВЧАЛЬНИЙ ПОСІБНИК

Відповідальний за випуск Бутенко В.М.

Редактор Еткало О.О.

Підписано до друку 10.06.09 р.

Формат паперу 60х84 1/16 . Папір писальний.

Умовн.-друк.арк. 11,75. Обл.-вид.арк. 12,0.

Замовлення № Тираж 500. Ціна

Видавництво УкрДАЗТу, свідоцтво ДК № 2874 від. 12.06.2007 р.
Друкарня УкрДАЗТу,
61050, Харків - 50, пл. Фейєрбаха, 7

Варіант	a1	a2	Номер рядка / стовпця	найбільшим/ наименшим	додатних/ від'ємних	парних/ непарних	рядках/ стовпцях	a4		a5		по спаданню/ по зростанню	вище/ нижче
1	-15	25	рядка 1	Першим НАЙБІЛЬШИМ	Додатних	парних	рядках	-13	-8	5	19	по спаданню	вище
2	-6	40	стовпця 1	Першим НАЙМЕНШИМ	Від'ємних	непарних	стовпцях	-4	1	27	34	по зростанню	нижче
3	-40	10	рядка 2	Останнім НАЙБІЛЬШИМ	Додатних	непарних	рядках	-37	-32	-34	4	по спаданню	нижче
4	-16	25	стовпця 2	Останнім НАЙМЕНШИМ	Від'ємних	парних	стовпцях	-13	-8	5	19	по зростанню	вище
5	-10	40	рядка 3	Першим НАЙБІЛЬШИМ	Додатних	непарних	рядках	-7	-2	26	34	по спаданню	вище
6	-42	10	стовпця 3	Першим НАЙМЕНШИМ	Від'ємних	парних	стовпцях	-39	-34	-24	4	по зростанню	нижче
7	-15	25	рядка 4	Останнім НАЙБІЛЬШИМ	Додатних	парних	рядках	-11	-6	7	19	по спаданню	нижче
8	-6	40	стовпця 4	Останнім НАЙМЕНШИМ	Від'ємних	непарних	стовпцях	1	6	20	32	по зростанню	вище
9	-40	10	рядка 5	Першим НАЙБІЛЬШИМ	Додатних	парних	рядках	-33	-30	-15	2	по спаданню	вище
10	-16	25	стовпця 5	Першим НАЙМЕНШИМ	Від'ємних	непарних	стовпцях	-9	-4	7	17	по зростанню	нижче
11	-10	40	рядка 1	Останнім НАЙБІЛЬШИМ	Додатних	непарних	рядках	-6	-1	25	32	по спаданню	нижче
12	-42	10	стовпця 2	Останнім НАЙМЕНШИМ	Від'ємних	парних	стовпцях	-41	-36	-28	2	по зростанню	вище
13	-15	25	рядка 4	Першим НАЙБІЛЬШИМ	Додатних	непарних	рядках	-11	-6	10	17	по спаданню	вище
14	-16	40	стовпця 5	Першим НАЙМЕНШИМ	Від'ємних	парних	стовпцях	-14	-9	31	34	по зростанню	нижче
15	-44	10	стовпця 3	Останнім НАЙБІЛЬШИМ	Додатних	парних	рядках	-42	-37	-27	4	по спаданню	нижче

ЗМІСТ

ВСТУП

Навчальна дисципліна "Обчислювальна техніка, програмування, моделювання систем" входить у навчальний план підготовки бакалаврів по галузі знань 0701 «Транспорт і транспортні технології». Посібник розрахований для вивчення трьох модулів зазначеної дисципліни, а саме: «Алгоритмічні мови програмування», «Програмування однорідних та різнорідних структурованих даних» та «Додаткові можливості QBasic».

Мета дисципліни – ознайомити студентів з основами програмування та сучасними алгоритмічними мовами. Зокрема у курсі розглядаються основні конструкції QBasic, аналізуються основні типи й структури даних, висвітлюються питання об'єктно-орієнтованого програмування на Visual Basic 6.0 (VB6), дається стислий огляд технології розробки програмних проектів.

Навчавльний посібник "Основи програмування мовами високого рівня" є логічним продовженням посібника "Основи алгоритмізації базових обчислювальних процесів", написаного колективом викладачів кафедри обчислювальної технки та систем управління УкрГАЖТ та надрукованого в академії у 2008 р.

"Універсальна мова для навчання початківців" – так визначили призначення алгоритмічної мови винахідники Basica Дж. Кемені й Т. Куртс. Basic продемонстрував за 37 років свого розвитку якості, властиві міфічному Протею. Здатність середовища цієї мови приймати десятки різних облич дозволяє вважати його своєю мовою й фізикам, і економістам, і механікам, і транспортникам, і не в останню чергу програмістам-професіоналам. Глави даної книги прагнуть навчити всіх бажаючих програмуванню і роблять це по можливості на нескладних і зрозумілих прикладах.

Мета видання посібника – навчити програмуванню початківців і дати деякі корисні поради тим, хто вже вміє програмувати. Більш конкретна третя ціль – навчити програмувати засобами QBasic і VB6.

Щоб пояснити, чому перевага віддається саме Basicy, будуть потрібні, мабуть, занадто великі для введення міркування. Питання про сильні й слабкі його сторони в порівнянні з мовами Сі, Паскаль, про місце його у сімействі мов програмування будуть порушені на сторінках книги. На думку авторів, на сьогоднішній день для здійснення перших двох цілей книги найбільше підходять вищезгадані алгоритмічні мови. Вони прості в освоєнні, широко поширені, не вимагають великих

ресурсів машини, відмінно працюють на всіх моделях ПК. За допомогою них можна ефективно вирішувати досить широке коло задач, а при належному підході вони здатні прищепити гарні програмістські навички, які стануть в нагоді при освоєнні будь-яких інших мов програмування.

Ще більше піднімає Basic як базову мову програмування те, що такі популярні додатки Windows, як Word та Excel, використовують його як засіб для розширення своїх можливостей.

Особливість книги в тому, що вона одночасно є своєрідним посібником з програмування для початківців і в той же час – посібником з QBasic і VB6. Іноді переважає перша тенденція, іноді – друга. У кожному разі автори прагнули до максимально повного викладення можливостей цих програмних засобів, але при цьому все-таки орієнтуючись більше на початківців (хоча й не завжди).

Обраний авторами книги стиль викладення прийдеться до душі тим, хто прагне реалізувати в рішенні своїх задач глибинні можливості обраної мови програмування. Надаючи різноманітні можливості для застосування різних прийомів ручної оптимізації коду й просто програмістських трюків, Basic допомагає розвивати уяву й почуття стилю, що так необхідно програмістам за всіх часів. Саме ці конструктивні можливості Basica компенсують його недоліки, які згадують багаторазово в літературі і які, до речі, легко можна обійти.

У фільмі "Космічна Одиссея 2001" по книзі Артура Кларка бортовий комп'ютер корабля "Дискавери" розмовляє з астронавтом на Фортрані. Чесно кажучи, сьогодні це здається малоймовірним. Чи не дістанеться ця роль у реальності черговому діалекту Basica?... Таке враження, що в його реалізаціях, що змінюють одна іншу, є якась загальна модель імунітету, що дозволяє їм жити й розвиватися під розмови про чарівні властивості Ади та Прологу. Займіться Basicом - він дозволить вам показати, на що Ви здатні.

Basic – молодший за рангом і одночасно один з найстарших у сімействі мов програмування. Автори намагались перебороти багато протиріч різних його описів, обумовлені тривалим періодом розвитку мови й участю в його розробці великої кількості фахівців, що часто належать різним поколінням. Їм довелося ретельно розібрати й систематизувати нашарування, придбані мовою за довгі роки "життя", і створити компас для орієнтування в його численних можливостях. Багато тонкощів мови вдалося виявити тільки експериментальним шляхом,

"запитуючи" у його інтерпретатора те, що не освітлено ні в публікаціях, ні в довідковій системі. Вважаючи на те, що книга в першу чергу адресована новачкам у програмуванні, практично всі можливості мови ілюструються прикладами, закріплюються питаннями для самоконтролю й практичними завданнями.

Для роботи з даним посібником потрібен певний рівень володіння комп'ютером. Передбачається, що Ви вмієте виконувати найпростіші завдання, наприклад запускати програми й переміщатися по файловій системі комп'ютера за допомогою провідника Windows.

Ви довідаєтеся, як організувати введення й виведення даних, як використовувати в програмах змінні, константи й коментарі, познайомитеся з основними правилами використання чисел і символьних рядків, керуючими конструкціями й циклами. Ви довідаєтеся, що таке підпрограми і яким чином вони спрощують написання великих програм, як створювати графічні зображення й змусити комп'ютер відтворювати звуки, як зберігати дані у файлах, а також як створити інтерфейс користувача й налагодити роботу програми. Окремі частини книги присвячені різним структурам даних. Легкий і доступний стиль викладу допоможе новачкам якнайшвидше почати створювати власні програми.

В главах, присвячених Visual Basic 6.0, розглянуті стандартні елементи для розробки інтерфейсу користувача, а також застосування в проектах таймерів, повзунків, гіперпосилань і інших елементів інтерфейсу. Приділено увагу основним поняттям об'єктно-орієнтованого програмування. Використані такі прийоми програмування, як методи, змінні, керування виконанням коду й створення власних класів - основних структурних елементів комп'ютерної програми, написаної на VB6.

Про що ви довідаєтеся ще? По-перше, ви можете відкрити для себе, що програмування – це захоплююче заняття! Коли завершуєш роботу над програмою і її поводження відповідає бажаному, то відчуваєш величезне почуття задоволення – чи то вигадана вами комп'ютерна гра, програма керування роботом або щось ще. На шляху до мети можуть виникати перешкоди – як і в будь-якій непростій справі, у програмуванні неминучі ускладнення – але коли бачиш результат своєї праці, то залишається тільки пишатися тим, що переборов всі випробування й втілив задумане в життя.

Крім того ми сподіваємося, що ви відкриєте для себе захопливість програмування, коли ви створите свою найпершу програму.

ТЕОРЕТИЧНІ ОСНОВИ ПРОЕКТУВАННЯ ПРОГРАМ ЗАСОБАМИ СИСТЕМИ ПРОГРАМУВАННЯ QBASIC

До середини 60-х рр. комп'ютери були занадто дорогими машинами, що використовувалися тільки для особливих завдань і виконували тільки одне завдання за раз (так звана пакетна обробка).

Мови програмування цієї ери, як і комп'ютери, на яких вони використовувалися, були розроблені для специфічних задач, зокрема для наукових обчислень. Однак протягом 60-х рр. ціна на комп'ютери почала падати так, що навіть невеликі компанії могли їх собі дозволити; швидкість комп'ютерів усе збільшувалася і настав час, коли вони часто простоювали без завдань. Щоб цього не відбувалося, ввели систему з поділом часу (time-sharing).

У таких системах процесорний час «нарізався», і всі користувачі по черзі одержували короткі відрізки цього часу. Машини були досить швидкими для того, щоб у результаті кожний користувач за терміналом почував себе так, начебто працює із системою поодинці. Машина ж у свою чергу простоювала менше, оскільки виконувала не одне, а відразу багато завдань. Поділ часу радикально знижував вартість машинного часу, оскільки одна машина могла спільно використовуватися сотнями користувачів.

У цих умовах – коли потужність стала дешева і доступна - творці мов програмування усе більше почали замислюватися про зручність написання програм, а не тільки швидкість їхнього виконання. «Дрібні» (атомарні) операції, виконувані безпосередньо пристроями машини, об'єднали в більш «великі», високорівневі операції і цілі конструкції, з якими людині куди простіше і зручніше працювати.

Із часу створення перших машин, що програмувалися, людство вигадало вже більше двох з половиною тисяч мов програмування. Щороку їхнє число поповнюється новими. Деякими мовами вміє користуватися тільки невелике число їхніх власних розробників, інші стають відомі мільйонам людей. Професійні програмісти іноді застосовують у своїй роботі більше десятка різноманітних мов програмування.

Для вирішення більшості завдань можна використовувати будь-яку мову програмування. Досвідчені програмісти знають, яку мову краще використовувати для вирішення кожної конкретної задачі, тому що кожна з мов має свої можливості,

орієнтацію на певні типи завдань, свій спосіб опису понять і об'єктів.

Мова програмування - формальна знакова система, призначена для опису алгоритмів у формі, що зручна для виконавця (наприклад, комп'ютера). Мова програмування визначає набір лексичних, синтаксичних і семантичних правил, використовуваних при складанні комп'ютерної програми. Вона дозволяє програмістові точно визначити те, на які події буде реагувати комп'ютер, як будуть зберігатися й передаватися дані, а також які саме дії варто виконувати над цими даними в різних умовах.

Творці мов по-різному тлумачать поняття *мова програмування.* Серед спільних місць, визнаних більшістю розробників, є такі:

- ***функція***: мова програмування призначена для написання комп'ютерних програм, які застосовуються для передачі комп'ютеру інструкцій з виконання того або іншого обчислювального процесу й організації керування окремими пристроями;
- ***задача***: мова програмування відрізняється від природних мов тим, що призначена для передачі команд і даних від людини комп'ютеру, у той час як природні мови використовуються лише для спілкування людей між собою. У принципі, можна узагальнити визначення "мов програмування" - це спосіб передачі команд, наказів, чіткого керівництва до дії; тоді як людські мови служать також для обміну інформацією;
- ***виконання***: мова програмування може використовувати спеціальні конструкції для визначення та маніпулювання структурами даних і керування процесом обчислень;

Процес роботи комп'ютера полягає у виконанні програми, тобто набору цілком певних команд у цілком певному порядку. Машинний вигляд команди, що складається з нулів і одиниць, указує, які саме дії повинен виконати центральний процесор. Виходить, щоб задати комп'ютеру послідовність дій, які він повинен виконати, потрібно задати послідовність двійкових кодів відповідних команд. Програми в машинних кодах складаються з тисячі команд. Писати такі програми - заняття складне й стомлююче. Програміст повинен пам'ятати комбінацію нулів і одиниць двійкового коду кожної програми, а також двійкові коди адреси даних, використовуваних при її виконанні. Набагато простіше написати програму на якій-небудь мові, більш близькій

до природньої людської мови, а роботу з перекладу цієї програми в машинні коди доручити комп'ютеру.

Усі мови програмування можна поділити на дві групи: мови низького рівня і мови високого рівня.

До мов низького рівня відносяться мови асемблера (від англійського *to assemble* – збирати, компонувати). У мові асемблера використовуються символьні позначення команд, які легко зрозумілі й швидко запам'ятовуються. Замість послідовності двійкових кодів команд записуються їхні символьні позначення, а замість двійкових адрес даних, використовуваних при виконанні команди, - символьні імена цих даних, обрані програмістом. Іноді мову асемблера називають мнемокодом або автокодом.

Більшість програмістів користуються для складання програм мовами високого рівня. Як і звичайна людська мова, така мова має свій алфавіт - безліч символів, використовуваних у мові. Із цих символів складаються так звані ключові слова мови. Кожне із ключових слів виконує свою функцію, так само як у звичній нам мові слова, складені з букв алфавіту даної мови, можуть виконувати функції різних частин мови. Ключові слова зв'язуються одне з одним у речення за певними синтаксичними правилами. Кожне речення визначає деяку послідовність дій, які повинен виконати комп'ютер.

Мова високого рівня виконує роль посередника між людиною й комп'ютером, дозволяючи людині спілкуватися з комп'ютером більш звичним для неї способом. Часто така мова допомагає обрати правильний метод вирішення задачі.

Перед тим, як писати програму мовою високого рівня, програміст повинен скласти алгоритм вирішення задачі, тобто покроковий план дій, який потрібно виконати для її вирішення. Тому мови, що вимагають попереднього складання алгоритму, часто називають алгоритмічними мовами.

Технологія програмування – це сукупність методів і засобів розроблення (написання) програм та порядок застосування цих методів і засобів.

На ранніх етапах розвитку програмування, коли програми писалися у вигляді послідовностей машинних команд, будь-яка технологія програмування була відсутня. Перші кроки в розробленні технології полягали в поданні програми у вигляді послідовності операторів.

Написанню послідовності машинних команд передувало складання операторної схеми, що відображає послідовність операторів і переходи між ними. Операторний підхід дозволив

розробити перші програми для автоматизації складання програм – так звані складові програми.

Зі збільшенням розмірів програм почали виділяти їхні відособлені частини й оформлювати їх як підпрограми. Частина таких підпрограм поєднувалася в бібліотеки, з яких підпрограми можна було включати в робочі програми і потім викликати з робочих програм. Це поклало початок процедурному програмуванню - велика програма подавалася сукупністю процедур-підпрограм. Одна з підпрограм була головною, з неї, як правило, починалося виконання програми.

Процедурний підхід вимагав структурування майбутньої програми, поділу її на окремі процедури. При розробленні окремої процедури про інші процедури було потрібно знати тільки їхнє призначення та спосіб виклику.

З'явилася можливість переробляти окремі процедури, не зачіпаючи іншої частини програми, скорочуючи при цьому витрати праці і машинного часу на розроблення й модернізацію програм.

Наступним кроком у поглибленні структурування програм стало так зване структурне програмування, при якому програма в цілому і в окремих процедурах розглядалася як послідовності канонічних структур: лінійних ділянок, циклів і розгалужень. З'явилася можливість читати й перевіряти програму як послідовний текст, що підвищило продуктивність праці програмістів при розробленні та відлагодженні програм. З метою підвищення структурності програми було висунуто вимоги до більшої незалежності підпрограм, підпрограми повинні зв'язуватися із програмами, що їх викликають, тільки шляхом передачі їм аргументів; використання в підпрограмах змінних, приналежних іншим процедурам або головній програмі, стало вважатися небажаним.

Процедурне й структурне програмування стосувалося насамперед процесу опису алгоритму як послідовності кроків, що ведуть від вихідних даних, що варіюються, до шуканого результату. Для вирішення спеціальних задач почали розроблятися мови програмування, орієнтовані на конкретний клас задач: на системи керування базами даних, імітаційне моделювання і т.і.

При розробленні трансляторів усе більше уваги почали приділяти виявленню помилок у вихідних текстах програм, забезпечуючи цим скорочення витрат часу на відлагодження програм.

Застосування програм у найрізноманітніших галузях людської діяльності привело до необхідності підвищення надійності всього програмного забезпечення.

Одним з напрямків удосконалювання мов програмування стало підвищення рівня типізації даних. Теорія типів даних виходить із того, що кожне використовуване в програмі дане належить одному і тільки одному типу даних. Тип даного визначає безліч можливих значень даного і набір операцій, припустимих над цим даним. Дане конкретного типу в ряді випадків може бути перетворене в дане іншого типу, але таке перетворення повинне бути явно подане в програмі. Залежно від ступеня виконання перерахованих вимог можна говорити про рівень типізації тієї або іншої мови програмування.

Усі універсальні мови програмування, незважаючи на розходження в синтаксисі та використовуваних ключових словах, реалізують ті самі канонічні структури: оператори присвоювання, цикли й розгалуження. У всіх сучасних мовах наявні визначені (базові) типи даних (цілі й речовинні арифметичні типи, а також символьний тип), є можливість використання агрегатів даних, у тому числі масивів і структур (записів). Для арифметичних даних дозволені звичайні арифметичні операції, для агрегатів даних звичайно передбачена тільки операція присвоювання й можливість звертання до елементів агрегату.

Разом з тим при розробленні програми для вирішення конкретної прикладної задачі бажана можливо більша концептуальна близькість тексту програми до опису завдання.

Вирішення цієї проблеми можливе декількома шляхами:

- побудовою мови програмування, що містить якнайбільше типів даних, і вибором для кожного класу задач деякої підмножини цієї мови. Таку мову іноді називають мовою-оболонкою;
- побудовою розширюваної мови, що містить невелике ядро і допускає розширення, яке доповнює мову типами даних і операторами, що відображають концептуальну сутність конкретного класу задач.

1.1. СЕРЕДОВИЩЕ QBASIC

1.1.1. Загальні відомості про середовище Qbasic

Назва мови програмування Basic – це перші літери англійських слів **B**eginner's **A**ll - purpose **S**ymbolic **I**nstruction **C**ode (багатоцільова мова програмування для початківців). Час створення першої версії – 1964 р.

У 1981 р. з'явилася розширена версія Basic-A, що підтримувала текстовий і графічний режими.

Розвитком Basic-A стала версія Quick-Basic, що включала підпрограми і функції з локальними й глобальними змінними, засоби підтримки графіки і звуку, алфавітно-цифрові мітки і т.ін.

Усіченим варіантом Quick-Basic є система програмування QBasic – інтегрована система програмування, що має власну управляючу оболонку, текстовий редактор, засоби пуску, відлагодження і потужний електронний довідник із системи.

Введений за допомогою екранного редактора текст програми необхідно перетворити в машинний код, що складається із двійкових даних та інструкцій процесора. Таке завдання вирішує спеціальна програма - транслятор. Найбільш простий спосіб трансляції - це інтерпретація (тлумачення), що полягає в негайному перекладі в машинні коди кожного слова програми.

Інтерпретатор QBasic може працювати у двох режимах: безпосередньому і режимі виконання програми.

У безпосередньому режимі дія, задана службовим словом, виконується відразу ж після його введення з клавіатури.

При роботі в режимі виконання текст програми спочатку повністю записується операторами мови, далі програма заноситься до пам’яті і запускається на виконання. У цьому випадку інтерпретатор слово за словом зчитує з пам'яті текст програми, перекладає його на мову команд процесора і пропонує готові коди процесору для їхнього виконання.

В інтерпретатора є велика перевага – простота відлагодження програм: виконання програми можна в будь-який момент зупинити і відредагувати її текст. Але є й істотні недоліки: по-перше, програма здатна працювати тільки при наявності в пам'яті самого інтерпретатора (середовища) і по-друге, програма, що працює з інтерпретатором, виконує всі дії дуже повільно. Час іде на розпізнавання слів мови, на пошук відповідних послідовностей команд процесора і т.ін. Ці недоліки усуваються за допомогою іншого способу трансляції – компіляції (зборки): дії не виконуються негайно, а збираються з наявних у компіляторі стандартних послідовностей у програму в кодах, що не залежить від яких-небудь трансляторів. Головне в компіляторі – зробити програму максимально швидкодіючою.

QBasic має тільки інтерпретатор, а от Quick-Basic дає можливість компілювати програми – створювати файли, що виконуються (exe-файли).

Після запуску QBasic на екрані з’являється перше вікно (рис. 1).

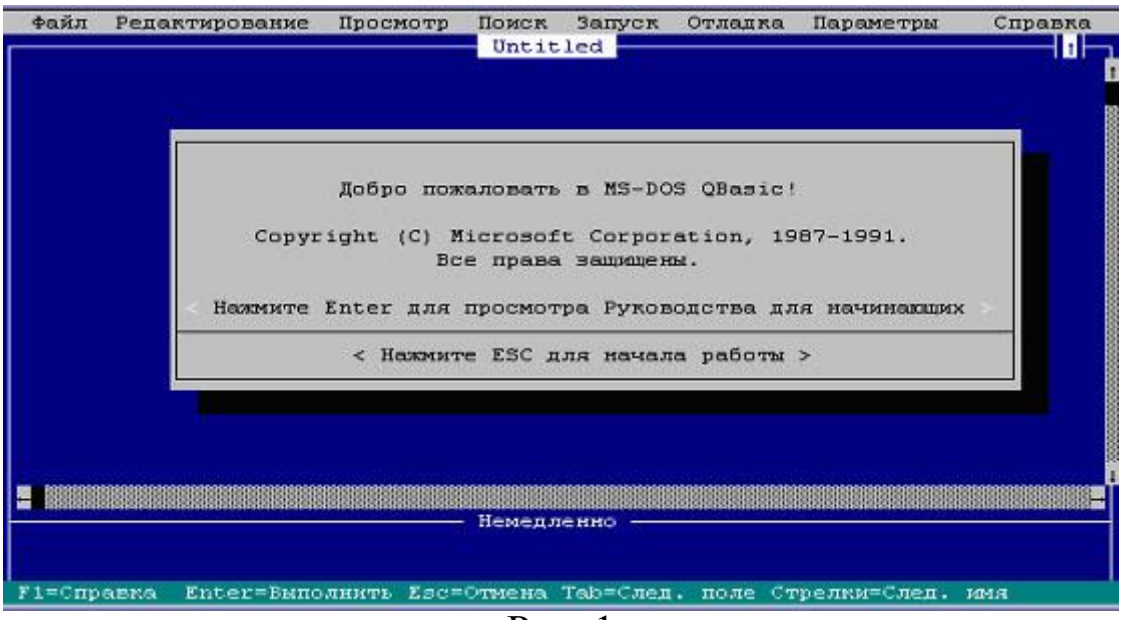


Рис. 1

Система меню

Головне меню QBASIC розташовано у верхній частині екрана. Воно містить назви основних функцій середовища.

Здійснити певні дії можна за допомогою таких функціональних клавіш та їх сполучень (табл. 1):

Таблиця 1

Сполучення клавіш	Дії, які виконуються
1	2
F1	Виведення підказки для того елемента програми, на який указує курсор
F2	Виведення списку підпрограм. Розподіл екрана на дві частини для одночасного перегляду великих програм
F3	Пошук у тексті
F4	Перегляд екрана виведення
F5	Продовження виконання програми з поточного оператора
F6	Переміщення у вікно швидкого виконання (Immediate)
F7	Виконання програми до поточного положення курсора
F8	Виконання наступного оператора програми (покрокове виконання)
F9	Установка або видалення контрольної точки
F10	Виконання наступного оператора програми (пропускаючи процедуру)

Продовження таблиці 1

1	2
Shift+F1	**Перехід у режим довідки**
Shift+F2	**Перехід до наступної процедури**
Shift+F5	**Виконання програми з початку**
Shift+F6	**Перемикання у вікно перегляду**
Ctrl+F1	**Перегляд наступної теми в режимі довідки**
Ctrl+F2	**Перехід до попередньої процедури**
Ctrl+F10	**Перемикання між багатовіконним і повноекранним режимами**
Alt+F1	**Перегляд попередньої теми в режимі довідки**

Введення і редагування програм

Робоча область або головне вікно QBasic розділена на дві частини – вікно редагування й вікно безпосереднього виконання. **Вікно редагування** призначене для введення тексту нової програми (Опція Головного меню **Файл** ⇒ команда **Новый**), завантажування текстів існуючих програм (**Файл ⇒ Открыть ⇒** клавіша **Tab** – вибрати зі списку), редагування введених текстів (**Редактирование**), запуску програми на виконання (**Запуск ⇒ Запуск**), зберігання їх на диску (**Файл ⇒ Сохранить**).

Вікно безпосереднього виконання.

Розташоване в нижній частині екрана. Призначене для того, щоб безпосередньо одержувати результати виконання команди після її введення і натискання клавіші **Enter**.

Редактор QBasic

Якщо при введенні оператора ви допустили помилку, то її можна виправити, використовуючи клавіші переміщення курсора і клавіші видалення символу:

- **Delete** – видаляє символ над курсором;
- **Backspace** – видаляє символ ліворуч від курсора.

При наборі застосовують також клавіші:

- **Shift** – перемикач режимів введення великих літер (верхній регістр) і малих літер (нижній регістр);
- **Caps Lock** – фіксація режиму введення великих літер і скасування його.

Текст програми можна вводити в будь-якому регістрі. Однак після натискання клавіші **Enter** всі службові слова програми будуть виведені великими літерами.

Робота із блоками тексту

Дуже корисним є використання блоків тексту. Це підвищує швидкість введення програм і відповідно зменшує вірогідність помилок. Існують такі можливості:

- виділення блоку;
- копіювання;
- переміщення;
- видалення.

Тільки виділений текст можна копіювати, перемістити або видалити. Виділений фрагмент тексту легко відрізняється від решти тексту більш яскравим підсвічуванням.

Існують **кілька способів виділення тексту**:

- невеликий блок (слово, кілька слів, рядок, кілька рядків, екран) зручніше виділити одночасним натисканням **Shift** і однієї із клавіш керування курсором ↑, ↓, ←, →, **Home, End, PgUp, PgDn**. Зняти виділення можна натисканням миші на невиділеній частині тексту. Якщо не все виділене або прихоплене зайве і вже відпущена клавіша керування – не відпускайте **Shift** і натисніть клавішу керування у зворотньому напрямку. Комбінації **Ctrl**+**Shift**+← та **Ctrl** +**Shift**+→ виділяють слово ліворуч і праворуч від курсора.
- комбінації **Shift+Ctrl+Home** і **Shift+Ctrl+End** дозволяють виділити текст програми, відповідно від початку до поточного рядка і від кінця до поточного рядка. Спочатку натискується й утримується **Shift**, а потім послідовно натискаються й відпускаються інші клавіші.

Натисканням клавіші **Del** виділений блок видаляється без запам’ятовування в буфері обміну. Комбінацією клавіш

Ctrl+ Del – виділений блок зберігається в буфері.

Виділений блок можна скопіювати в буфер сполученням **Ctrl + Ins**, після чого сполученням **Shift + Ins** його можна вставляти в будь-які місця скількі завгодно раз у будь-які програми;

- все вищеназване можна робити за допомогою опцій меню

Редактирование (рис. 2).

Запуск програми на виконання і перегляд результатів

Коли текст програми набраний, її треба виконати за допомогою команди **Запуск** з меню **Запуск** або використати комбінацію клавіш **Shift+F5**.

На екран буде виведений результат виконання програми і повідомлення: **Press any key to continue** (Для продовження натисніть будь – яку клавішу).

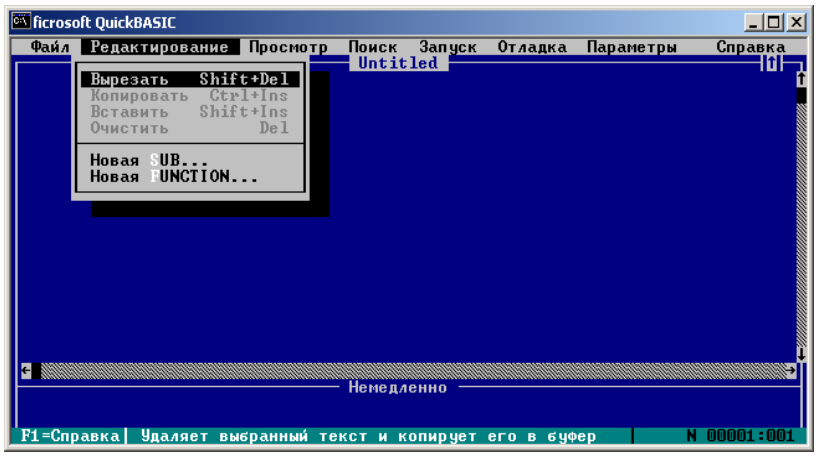


Рис.2

Відлагодження програм

Транслятор QBasic виявляє помилки двох типів:

- синтаксичні, які виникають в результаті порушення правил написання конструкцій мови;
- семантичні, пов'язані з неприпустимими значеннями параметрів, неприпустимими діями над параметрами і т.ін.

При виявленні помилки видається відповідне повідомлення на екран монітора і відбувається підсвічування місця її знаходження в тексті.

Крім того, на етапі безпосереднього виконання відслідковуються деякі помилкові ситуації: ділення на нуль, обчислення квадратного кореня з від'ємного числа, переповнення – вихід за праву границю діапазону типу числа і ряд інших. У цих випадках виконання програми або припиняється, або триває з видачею повідомлення про помилку.

Помилки в логіці роботи алгоритму не обов'язково приводять до зупинки програми. Виявити їх можна тільки аналізом проміжних і остаточних результатів. Для покрокового аналізу проміжних результатів у текст рекомендується включати виведення таких результатів, практикувати виключення фрагментів програми з її роботи шляхом перетворення їх у коментар, використовуючи апостроф.

Переглянути результати можна натиснувши **F4**.

Зупинити виконання програми можна натиснувши **Ctrl+Break**, а продовжити – натиснувши **F5**.

Для призупинення виконання програми можна поставити у відповідному місці оператор **STOP**.

Значно легше виявити логічні помилки, використовуючи спеціальні клавіші або опцію Головного меню **Отладка**.

Іноді виникає необхідність у виконанні програми до рядка, зазначеного курсором. Для цього треба натиснути **F7**. Аналогічні

результати досягаються натисканням клавіші **F9 –** установкою в опції **Отладка Контрольної точки** останова программы (вона ж – зняття цього режиму).

Після натиснення клавіші **F8** або **F10** (команди **Шаг**, **Процедура на шаг**) реалізується тільки один оператор, що дозволяє виконувати програму крок за кроком.

Команда **Трассировка** дозволяє відстежити послідовність виконання операторів програми.

Команда **Установить следующее значение** використовується для зміни послідовності виконання програми так, що наступним виконується оператор, на якому встановлений курсор.

Збереження програми на диску

Для зберігання набраної програми на диску у вигляді файла треба виконати такі дії:

Файл ⇒ Сохранить ⇒ Enter

Якщо програма ще не має імені, то вона в середовищі QBasic буде позначена як **Untitled**. Введіть ім'я файла, з яким ви бажаєте зберегти свою програму, і натисніть клавішу **Enter.**

Ім'я файла повинне бути унікальним. Рекомендується використати для цієї мети послідовність літер та цифр довжиною до 8 символів. Розширення файла **.bas** буде записане автоматично.

Довідкова система QBasic

У вікні **СПРАВКА** можна побачити таку інформацію:

- теми довідки;
- основні ключові слова.

Щоб одержати довідку по ключовому слову QBasic, встановіть на ньому курсор і натисніть **F1** або праву кнопку миші.

Щоб одержати довідку по меню, командам або діалоговому вікну QBasic, встановіть курсор на темі меню або на кнопці **СПРАВКА** і натисніть **F1**.

Щоб переглянути теми **СПРАВКА** QBasic, натисніть **Alt+З**, для вибору теми – натисніть букву, виділену підсвічуванням.

Щоб перемістити курсор у вікно довідки, натисніть **Shift+F6**.

Щоб переглянути всю довідкову інформацію, використайте **PgDn** або **PgUp**.

Щоб скопіювати інформацію з **СПРАВКА** (приклади програм) у вікно редагування, використайте команди з меню **Редактирование** QBasic.

Щоб закрити вікно довідки, натисніть **Esc**.

Щоб перемістити курсор до теми **СПРАВКА**, необхідно використати клавішу табуляції або натиснути першу літеру теми. Щоб побачити інформацію з теми або ключового слова, треба встановити курсор на темі або ключовому слові і натиснути **F1** або праву кнопку миші.

QBasic зберігає останні 20 тем довідки, які були переглянуті. Щоб переглянути попередні теми, треба натиснути **Alt+F1** або натиснути кілька разів кнопку миші на кнопці **Назад**.

Вихід Із Середовища QBasic

Для виходу із середовища QBasic необхідно виконати послідовно такі дії:

Файл ⇒ Выход ⇒ Enter

1.2. ЕЛЕМЕНТАРНІ КОНСТРУКЦІЇ МОВИ BASIC

1.2.1. Алфавіт

Основою будь-якої мови програмування є алфавіт:

Алфавіт – набір припустимих знаків, які можна використати для запису програм.

Він включає всі відображувані й керуючі символи, наведені в таблиці ASCII-кодів ПК. Кожному символу алфавіту відповідає індивідуальний числовий код у діапазоні від 0 до 255 (0-127 – основна таблиця, 128-255 – розширена).

Приклад. Символу «A»(латинське) відповідає код 065, «a»(латинське) – 097, «А»(кирилиця) – 128, «а»(кирилиця) – 160, символу «>» – 062, символу «*» – 015.

Символи з кодами від 0 до 31 називають керуючими. Серед них є символи, які не відображаються на екрані.

Приклад. Символ BEL з кодом 7 – «виведення звукового сигналу», символ LF з кодом 10 – «переведення рядка», символ CR з кодом 13 – «повернення каретки».

Всі символи можна розділити на групи:

- **латинські літери** (26 літер) (A-Z, a-z);
- **російські літери** (кирилиця) (А-Я, а-я);
- **цифри** (0-9).

Розділові символи (обмежники)

. десяткова крапка	**, кома**
: двокрапка	**« лапки**
( ) круглі дужки	**пропуск**
; крапка з комою	**‘ одинарні лапки (апостроф)**

Спеціальні знаки

$ знак грошової одиниці	**@ комерційне ЕТ**
& комерційне І (амперсанд)	**_ символ підкреслення**
# номер	**% знак відсотка**
? ! {} і т.ін.	

Знаки арифметичних операцій

*** множення**	**/ ділення (прямий слеш)**
^ піднесення до степеня	**+ додавання**
- віднімання	
MOD – залишок від цілочисельного ділення чисел	**\ цілочисленне ділення (зворотній слеш) – повертає кількість входжень другого цілого числа в перше**

Знаки операцій відношення

Основні	**Похідні операції**
> знак більше ніж	**>= знак більше або дорівнює**
< знак менше ніж	**<= знак менше або дорівнює**
= знак дорівнює	**<> знак не дорівнює**

1.2.2. Службові слова і команди

Програма мови програмування високого рівня складається з окремих лексичних одиниць – команд (операторів).

Оператор – основна мінімальна логічно завершена конструкція мови, що містить ім’я оператора і його параметри. Оператор визначає об’єкти, тип і послідовність дій над ними. Для написання операторів і найменувань різноманітних об’єктів використовуються латинські літери.

Кирилиця використовується тільки для запису символьних констант і коментарів.

QBasic не робить різниці між великими і малими латинськими буквами в службових словах, позначеннях змінних і інших об’єктів.

За призначенням оператори розділяють на такі групи:

- описові оператори – для опису даних, типів змінних, розмірів масивів, нестандартних функцій і т.ін.;

• оператори присвоювання – для завдання початкового значення або зміни поточного значення змінної;

• оператори введення-виведення даних;

• оператори управління процесом обробки інформації- для організації розгалужень, циклів і т.ін.;

• інші оператори, що забезпечують додаткові можливості- для роботи з файлами даних, для графічних побудов, для одержання звукових ефектів і т.ін.

Оператори складаються з нероздільних елементів мови: службових слів, чисел, символів операцій і т.п. QBasic налічує більше 200 типів операторів. Наведемо деякі з них.

DATA – дані	**NEXT – наступний**
DEF FN – визначення функції	**ON – при**
DIM – розмір	**OPTION BASE – задати базу**
END – кінець	**PLAY – грати**
FOR-TO-STEP – для – до – крок	**PRINT – друкувати**
GOSUB – перейти до підпрограми	**READ – читати**
GOTO – перейти до	**REM – пояснення**
IF-THEN – якщо-то	**RESTORE – обновити**
INPUT – ввести	**RETURN – повернутися**
LET- нехай	**STOP – зупинитися**

1.2.3. Структура QBasic-програми

Програма складається з програмних рядків довжиною до 255 символів. В одному рядку можна розмістити один або кілька операторів, розділених двокрапкою.

Рядок може мати номер – ціле число, що трактується системою як **мітка.** Мітка може бути і символьною, **наприклад, Label1**. Після символьної мітки ставиться двокрапка.

1.2.4. Типи даних

Залежно від того, чи змінюють свої значення дані в процесі виконання програми, розділяють постійні величини (константи) і змінні.

1.2.4.1. Константи

Константи – це об’єкти, значення яких не можуть бути змінені в процесі виконання програми.

Константи за типом поділяються на **числові**, **логічні** і **символьні**.

а) числові константи

По відсутності або наявності дробової частини і по способу зберігання в пам'яті ПК діляться на цілі і речовинні (дійсні).

Для подання чисел у пам'яті комп'ютера використовуються чотири формати:

- з фіксованою крапкою;
- із плаваючою крапкою

У форматі з фіксованою крапкою подаються тільки цілі числа, а у форматі із плаваючою крапкою – речовинні числа (цілі і дробові).

Діапазон значень залежить від розміру комірок пам'яті, де вони зберігаються.

В k-розрядній комірці може зберігатися 2^k різних значень цілих чисел.

Приклад. З урахуванням знака числа в 16 – розрядній комірці (2 байти) зберігаються цілі числа в діапазоні від -2^{k-1} (-32768) до $2^{k-1} – 1$(32767).

Двійкові розряди в комірці нумеруються від 0 до k. Старший, k-й розряд, у внутрішньому поданні будь-якого додатного числа – 0, від'ємного – 1.

Формат із плаваючою крапкою використовує подання дійсного числа (назвемо його **R)** у вигляді добутку мантиси **m** на основу системи числення **n** у деякому цілому ступені **p**, що називають порядком:

$$R=m*np$$

Подання в цій формі неоднозначне:

$$25.324=2.5324*10^1=0.0025324*10^4=2532.4*10^{-2}$$

В ПК використовується нормалізоване подання числа у формі із плаваючою крапкою. При цьому мантиса повинна задовольняти умові $0.1 \le m < 1$, тобто повинна бути менше 1 і перша значуща цифра не дорівнювати 0.

У пам'яті мантиса зберігається як ціле число, що містить тільки значущі цифри (0 цілих і крапка не зберігаються).

Приклад. В 4-байтовій комірці:

- знак числа (1 розряд): 0- плюс, 1 – мінус;
- порядок (7 розрядів): числа від 0000000 до 1111111 (від 0 до 127 у 10-річній системі числення; усього 128 значень). Порядок очевидно може бути як додатним, так і від'ємним

(від -64 до 63). Машинний порядок (**Мр**) зміщений щодо математичного (**р**) і приймає тільки додатні значення. Зсув вибирається так, щоб мінімальному математичному значенню відповідав 0: **Мр=р+64**

- мантиса (24 розряди).

+/ - маш.порядок	МАН	ТИ	СА
1 байт	2 байт	3 байт	4 байт

Цілі константи – послідовність цифр і спеціальних символів зі знаком або без нього.

За способом задавання в програмах розрізняють:

- цілі константи в **десятковому** поданні:

- одинарної точності (тип INTEGER) – належать діапазону від -32768 до 32767 і зображуються в пам'яті 2-байтовими двійковими числами. У програмі запис цих констант завершується символом % (цілі короткі).

Приклад. **3%, -19%.**

- подвійної точності (тип LONG) – належать діапазону від 2147483648 до 2147483647 і зображуються в пам'яті 4-байтовими двійковими числами. Запис цілочисельних констант цього типу завершується символом & (цілі довгі).

Приклад. **123897890&, -78989&;**

- цілі константи у **вісімковому** поданні:

послідовність вісімкових цифр без знака, яким передує &.

Приклад. **&113, &2;**

- цілі константи в **шістнадцятковому** поданні:

послідовність шістнадцяткових цифр без знака, яким передує &H.

Приклад. **&H123, &HA56E.**

Для запису речовинних констант використовують дві форми: природну й експоненційну.

Речовинна константа в природній формі – послідовність десяткових цифр зі знаком і крапкою, що розділяє цілу і дробові частини.

Приклад. **0.775; -15.04; +11.08.**

Речовинна константа в експоненційній (показовій) формі (з плаваючою крапкою) має вигляд:

$$\pm mE \pm p$$

Мантиса записується аналогічно запису відповідної константи в природній формі;

E (D – для подвійної точності) – символ ознаки порядку; значення порядку записується зі знаком або без нього.

Приклад.0. 61E+12, 16. 333D-2

0.11E+12 або 1.1E+11 відповідає $0.11*10^{12}$.

Ця форма зручна для запису дуже великих або дуже маленьких чисел.

Знак плюс можна опускати. Знак мантиси визначає знак числа. Порядок може бути тільки цілим числом: він визначає положення десяткової крапки.

Кажуть «крапка плаває», тому що розміщення десяткової крапки в мантисі явно не вказує на величину числа.

За обсяг зайнятої пам'яті виділяють такі речовинні константи:

- **речовинна константа одинарної то**чності (тип SINGLE)

- діапазон для додатних чисел: від 2.8E-45 до 3.4E+38; для від'ємних: від -3.4E+38 до -2.8E-45;

- містить не більше 7 цифр; в експоненційній формі записується з літерою E. Припустимо запис констант такого типу закінчувати символом '!'.

- для їх зберігання використовуються 4-байтові поля.

Приклад. **45.23, -.6,1.3E-04,29.5!**

- **речовинна константа подвійної точності** (тип DOUBLE)

- діапазон для додатних: від 4.9D-324 до 1.8D+308, для від'ємних – від -1.8D+308 до -4.9D-324

- містить не більше 18 цифр; в експоненційній формі записується з літерою D. Запис таких констант закінчується символом #.

- для їхнього зберігання використовуються 8-байтові поля.

Приклад. **234568.61, -1.0965434D12, 3241.6#;**

б) логічні константи

Вони можуть мати тільки два фіксованих значення TRUE (ІСТИНА) або FALSE (ХИБНІСТЬ);

в) символьні (рядкові, текстові) константи (тип STRING)

Укладена в лапки послідовність символів (літер, цифр, спеціальних знаків – усього, що можна набрати на клавіатурі). Максимальна довжина символьної константи – 32767 символів. Якщо заздалегідь відомо, що число символів у константі не перевищує **n**, то тип задається, як **STRING* n**.

Приклад. **« аудиторія 4. 111 «, «=БЮТ+»**

Константам можна присвоювати імена за допомогою команди оголошення констант:

CONST і*м.'я_константи*[**As** *тип_даних*]=*вираз*[,*ім.'я_константи* [**As** *тип_даних*]=*вираз*], …

Обчислюється значення *виразу,* і результат присвоюється константі.

Приклад. **CONST PI=3.1415926, A=« програмування»**

1.2.4.2. Змінні величини

Змінні – це об'єкти, значення яких заздалегідь не визначені і можуть змінюватися в процесі виконання програми.

Змінним можна присвоювати значення констант, інших змінних, результатів обчислень або вихідних даних.

Кожна змінна позначається унікальним ім'ям – **ідентифікатором**. Кожному ідентифікатору в пам'яті ПК відповідає цілком визначена адреса комірки, і зміна значення змінної зводиться до зміни вмісту відповідної області пам'яті.

Ідентифікатор може містити від 1 до 40 алфавітно-цифрових символів. Першим символом повинна бути літера. В ідентифікаторах використовуються тільки латинські літери. Допускається використання крапок. Ім'я не повинне збігатися з яким-небудь із зарезервованих службових слів QBASIC (DATA, END, THEN і т.д.).

Приклади ідентифікаторів. A, b12, Beta C_mod, Gamma, End.of.word.

Імена змінних визначають їхній тип, а також точність числових змінних.

Змінні називаються числовими, якщо вони можуть набувати тільки числових значень.

Як і константи, змінні за типом діляться на числові, логічні і символьні.

Тип змінних в програмі можна задати чотирма способами:

1. Принцип умовчання
2. Спеціальні символи
3. Команди опису типу DEF
4. Команди оголошення змінних DIM

Принцип умовчання

Якщо в програмі немає опису типу змінних, то будь-які змінні вважаються речовинними. Однак символьні дані потрібно описувати завжди.

Явне задавання типу змінної – за допомогою спеціального символу (%, &, !, #, $), що завершує ім’я змінної (табл. 2).

Таблиця 2

Тип	Спеціальний символ	Обсяг займаної пам'яті	Приклад
Цілочисловий короткий	**%**	**2**	**F%, HD2%**
Цілочисловий довгий	**&**	**4**	**F&, HD2&**
Речовинний нормальний (одинарний))	**! або без символу**	**4**	**F!, HD2**
Речовинний довгий (подвійний)	**#**	**8**	**F#, HD2#**
Символьний	**$**	**Довільний**	**Y$**

При позначенні змінної 2-байтового цілочислового типу використовується символ % наприкінці змінної.

Приклад. **F%, HD2%**

Така змінна може набувати будь-якого значення, що припустимо для цілої константи нормальної довжини. Для позначення змінної 4-байтного цілочислового типу використовують символ &.

Приклад. **F&, HD2&**

Речовинна змінна одинарної точності (4 байти) звичайно позначається ім'ям без спеціальних символів (іноді запис ідентифікатора закінчується символом **!**).

Приклад. **F2, HD2!**

Для позначення змінної речовинного типу подвійної точності (8 байтів) використовується символ # наприкінці ідентифікатора.

Приклад. **F2#, HD2#**

Логічна змінна - символічно позначена логічна величина, що може набувати тільки значення TRUE (ІСТИНА) або FALSE (ХИБНІСТЬ).

Символьна змінна закінчується символом **$**.

Приклад. Змінна Y$ може містити дані символьного типу.

Неявне завдання типу змінної (команди опису типу) - за приналежністю першого символу ідентифікатора заданому діапазону літер.

Воно здійснюється за допомогою операторів

DEFINT	**2-байтовий цілочисловий**
DEFLNG	**4-байтовий цілочисловий**
DEFSNG	**4-байтовий речовинний**
DEFDBL	**8-байтовий речовинний**
DEFSTR	**символьний**

Всі ці оператори мають однаковий формат:

DEF<*тип***> <***діапазон_літер***>**

де <*діапазон_літер*> - літера або діапазон літер латинського алфавіту, з яких може починатися ім'я змінної відповідного типу.

Приклад. **DEFINT A, P-S, Z**

Всі змінні, що починаються з A,P,Q,R,S,Z вважаються 2-байтовими цілого типу.

На змінні, описані явним способом, неявне оголошення типів змінних не поширюється.

Приклад. У програмі поряд з оператором DEFSNG A,X-Z можуть існувати і чисельні змінні з іменами **YGAS%, AVAS&** і т.п.

Неявний опис типу поширюється тільки на ті змінні, імена яких не завершуються зазначеними вище спеціальними символами.

Використання команди оголошення змінних DIM

Традиційно цей оператор використовується для оголошення масивів. Однак він може бути використаний і для скалярних змінних в такому вигляді:

DIM <*список1_змінних*> AS <*тип1*>…[<,*список n_змінних*> AS <*тип n*>]

Приклад. Цілі змінні **A,B**, речовинні **C,D** і текстові **F,H** можна описати

```
DIM  A,B AS INTEGER, C,D AS DOUBLE
DIM  F,H AS STRING*10
```

Перед виконанням програми всім числовим змінним автоматично присвоюються нульові значення. Довжина кожної змінної символьного типу встановлюється також нульовою, тобто всім символьним змінним присвоюється значення "порожнє".

Поряд з константами і змінними (скалярними) у програмах можуть використовуватися масиви і структури, тобто групи змінних, на які посилаються за одним загальним ім’ям. Окремі елементи таких груп використовуються в програмах за тими ж правилам, що і прості змінні.

1.2.4.3 Структури

Запис - структурований набір даних, призначений для зберігання в оперативній пам'яті й обробки даних, що складається з фіксованого числа компонентів даних різних типів (полів). Іноді записи називають структурою. Зрозуміло, типи компонентів можуть бути й однаковими.

Формат

TYPE *< ім'я запису>*
< ім'я поля 1> **AS** *< тип поля 1>*
.....
< ім'я поля n> **AS** *< тип поля n>*
END TYPE

Імена записам і полям дає користувач. Типом поля може бути кожний зі стандартних типів (INTEGER, LONG, SINGLE, DOUBLE, STRING*n) або інший запис. Типом поля не може бути тип невідомої довжини, тобто STRING.

Полями запису, крім стандартних числових типів даних і символьних змінних заданої довжини, можуть бути типи користувача. Для опису складних структур у записах як поля можна використати інші записи.

Приклад. Запис анкетних даних студентів: прізвище, ім'я, дата народження і середній бал, можна описати так:

```
TYPE grupa
    namel AS STRING*20
    surname AS STRING*15
    birthday AS TYPE
        year AS INTEGER
        month AS INTEGER
        day AS INTEGER
    END TYPE
    sball AS SINGLE
END TYPE
```

Змінну типу *запис* оголошують за допомогою команди

DIM *< ім'я змінної>* **AS** *< ім'я запису>*

Із записів можна створити масив.

Приклад. Оголосити змінну **a** і масив записів **stud** можна так:

```
DIM a AS grupa
DIM stud(20) AS grupa
```

Доступ до конкретного поля запису дає складене ім'я виду
< ім'я змінної>.< ім'я поля>

Приклад. У програмі змінним **a** і **stud(1)** можна задати такі значення:

a.name1="Петро"
stud(1).surname="Куваєв": stud(1).birtday.month=5

1.2.4.4. Функції

Як операнди у програмах поряд з константами, змінними й обумовленими користувачем функціями можна застосовувати функції, визначені в самій мові (вбудовані функції). На використання більшості з них не накладаються ніякі обмеження.

Для зручності всі функції поділяють на групи, зв'язані не тільки з типом функції, але і з її застосуванням і типом аргументів:

1) математичні функції;
2) числові функції символьних аргументів;
3) символьні функції;
4) функції введення, виведення і доступу до пам'яті;
5) системні змінні.

Математичні функції

У математичних функцій аргументами є арифметичні вирази, а значеннями - числа.

Таблиця 3

Стандартні математичні функції

Запис на Basic	Назва функції	Приклад
1	**2**	**3**
SIN(X)	**Синус X**	**sinX, аргумент в радіанах**
COS(X)	**Косинус X**	**cosX, аргумент в радіанах**
TAN(X)	**Тангенс X**	**tgX, аргумент в радіанах**
ATN(X)	**Арктангенс X**	**arctgX, $-\pi/2<X<\pi/2$,**
ABS(X)	**Модуль X**	**|X|**
SQR(X)	**Квадратний корінь X**	**$\sqrt{X}$, X>0**
EXP(X)	**Експонента X**	**e^X**
LOG(X)	**Натуральний логарифм X**	**lnX, X>0**
CDBL(X)	**Перетворення в число подвійної точності**	
CINT(X) ***CLNG(X)***	**округлення до цілого (довгого цілого) значення**	**CINT(15.5)=16** **CINT(-6.7)=-7** **CINT(-6.2)=-6**
CSNG(X)	**Перетворення в число простої точності**	

Продовження таблиціц 3

1	*2*	*3*
3FIX(X)	**Усікання до цілого (відкидання цілої частини)**	**FIX(-5.6)=-5 FIX(24.07)=24.00**
INT(X)	**Знаходження найбільшого цілого , що не перевищує X**	**INT(15.5)=15 INT(-6.2)=-7 INT(-6.7)=-7**
RND(X)	**Генерація псевдовипадкових чисел від 0 до 1**	
SGN(X)	**Знак числа X**	**SGN(X) = -1, якщо X < 0 SGN(0) = 0, якщо X = 0 SGN(X) = +1, якщо X > 0**

Числові функції символьних аргументів

Значеннями цих функцій є числа, а аргументами - символьні вирази або функції.

Приклад. ASC, CVI, CVS, CVD, INSTR, LEN, VAL та ін.

Символьні функції

Значеннями цих функцій є ланцюжки символів.

Приклад. CHR$, LEFT$, RIGHT$, MID$, STRING$, LTRIM$ та ін.

Всі функції 2 і 3-й груп в основному використовуються для перетворення даних, що поставляють оператори введення-виведення й обробки символьної інформації.

Функції введення - виведення і доступу до пам'яті

Сюди входять функції, пов'язані із введенням - виведенням, а також такі, що дозволяють одержати інформацію про стан різних пристроїв і транслятора QBasic.

Приклад. CSRLIN - повертає номер рядка поточного положення курсора.

FRE - обсяг вільної частини пам'яті робочої області QBasic.

POINT - координати точки екрана

PEEK - вміст байта пам'яті та ін.

Системні змінні

Приклад. TIME$ - системний час DATE$ - системна дата та ін.

Повністю список всіх функцій можна знайти у довідці QBasic і системній документації.

1.2.4.5. Вирази

Вираз - це послідовний запис констант, змінних, індексних змінних (елементів масивів), функцій або будь-яких їхніх комбінацій, утворений за допомогою знаків арифметичних і логічних операцій, операцій відношень, операцій над рядками символів.

Розрізняють арифметичні, логічні і символьні вирази.

Арифметичні вирази

Арифметичні вирази відповідають загальноприйнятим алгебраїчним виразам. До них можуть входити константи, змінні, функції, з'єднані знаками арифметичних операцій. Результатом арифметичного виразу є число. Число або змінна також вважаються арифметичним виразом. Для позначення арифметичних операцій використовуються знаки

+	-	*	/	\	MOD

Операція \ означає ділення націло (дробова частина відкидається)

Приклад. 17\2=8.

Операція **MOD** означає обчислення залишку (ділення по модулю)

Приклад. 25MOD8=1.

Якщо операнди цих операцій речовинні, то вони попередньо округлюються.

Приклад.

$\frac{ab}{c} = A * B / C$ $\qquad$ $\frac{x+2}{c+d} = (X+2)/(C+D)$

$\frac{a-\sqrt{d}}{2x} = (A - SQR(D))/(2 * X)$ $\qquad$ $\sin^2 x - \cos x^2 = SIN(X)\text{^}2 - COS(X\text{^}2)$

$3a^2 + \frac{b}{4} + \frac{7}{1-a} = 3 * A\text{^}2 + B/4 + 7/(1-A)$ $\qquad$ $\frac{y_j y_{j+1}}{\sqrt{j}} = Y(J) * Y(J+1) / SQR(J)$

Пріоритет виконання операцій

Всі операції в арифметичному виразі виконуються в певній послідовності: порядок їхнього виконання задається правилами пріоритету:

1) обчислення числових функцій;
2) унарний мінус;
3) піднесення до степені;
4) множення / ділення;
5) ділення націло;
6) обчислення залишку;
7) додавання / віднімання.

Послідовність операцій одного пріоритету виконується зліва направо. Для зміни цього порядку можна використати круглі дужки. Вирази, що знаходяться в дужках, обчислюються в першу чергу.

Приклад. **A =8 B =4 C =2.**

A+B/C^2 =9 (A+B)/C^2 - =3
(A+B/C)^2 =100 A+(B/C)^2 - =12

Якщо у виразі кілька операцій мають однаковий пріоритет, то вони виконуються одна за одною зліва направо. Винятком є піднесення до степеня

$$X^{Y^Z} = \textbf{X\^{}(Y\^{}Z).}$$

Для обчислення кореня довільного ступеня використовується еквівалентний вираз.

Приклад. $\sqrt[3]{\cos(x)}$ **=COS(X)^(1/3).**

Логарифм з довільною основою обчислюється за формулою

$$\log_a b = \frac{\ln b}{\ln a}.$$

Приклад. $\log_{10}(x+1)$ **=LOG(X+1)/LOG(10).**

Не можна ставити два знаки арифметичних операцій підряд, а треба використовувати дужки, **наприклад**, $\frac{a}{-b}$ =>A/(-B).

Від'ємні значення підносити до дробового степеня забороняється. Пояснюється це тим, що цей вираз перед обчисленням автоматично перетворюється як $x^a = e^{a \ln x}$.

У зв'язку з наближеним поданням речовинних чисел в ПК рівність **X/Y*Y=X** не виконується.

Відзначимо так звані особливі ситуації при обчисленні арифметичних виразів:

- ділення на нуль - видається відповідне повідомлення; обчислення тривають, в результаті буде **1.701412E+38-** машинна нескінченність;
- переповнення - видається відповідне повідомлення, і виконання програми припиняється.

Логічні вирази

Логічні вирази складаються з логічних операцій і логічних відношень. В них наявні два операнди, які QBasic розглядає як шістнадцяткові бінарні ланцюжки, над якими побітно зліва направо виконуються дії за допомогою таких операторів:

NOT	**заперечення (НІ)**	***XOR***	**АБО, що виключає**
AND	**кон'юнкція (І)**	***EQV***	**еквівалентність**
OR	**диз'юнкція (АБО)**	***IMP***	**імплікація**

Всі вони повертають значення ІСТИНА (не-нуль) або ХИБНІСТЬ (нуль), що використовуються при ухваленні вирішення про подальший хід обчислювального процесу.

Приклад. **63 AND 16=16** **4 OR 2=6**
111111 AND 10000=010000 **100 OR 010=110**

Порядок виконання логічних операцій задається пріоритетом (NOT, AND, OR) і круглими дужками.

Логічні відношення

Логічні відношення порівнюють два числових або символьних значення, і якщо результат ІСТИНА, виробляється - 1, інакше - 0.

Приклад. b2-4ac>0 B^2-4*A*C>0; i≠j I<>J;
рядок a = рядку b A$=B$;
46=41 результат 0; 10<8 результат 0; 15<=20 результат 1;
15<>20 результат 1; 2>1 результат 1; 3>=2 результат 1

Якщо порівняння робиться над числами, то припустиме порівняння цілого і речовинного типів.

Символьні величини можна порівнювати тільки із символьними, при цьому порівняння здійснюється посимвольно зліва направо із урахуванням кінцевих пропусків. Більш короткий рядок вважається меншим. Порівняння засноване на відносних значеннях їх шістнадцяткових кодів. При розбіжності символів більшим вважається рядок, що містить символ з більшим кодом.

Приклад. **"AVZ">"AGN"; код"V"=086; код"G"=071.**

Якщо логічні вирази містять логічні відношення і логічні операції, то спочатку виконуються логічні відношення, а потім логічні операції відповідно до пріоритету.

Приклад. **-4<= X <=4** **(-4<=X) AND (X<=4)**
X=0 або X=1 **(X=0) OR (X=1)**

Символьні вирази

Складаються із символьних констант, змінних, функцій або будь-яких їхніх комбінацій, розділених знаками логічних відношень або спеціальних символьних операцій.

Спеціальна символьна операція називається **конкатенація.** Вона символізує об'єднання значень і позначається символом "+ ".

Приклад. **A$="студент" B$=" гр. 5-І-В", тоді A$+B$="студент гр. 5-І-В"**

Довжина рядка, що є результатом, не повинна перевищувати 255 символів.

1.3. ОПЕРАТОРИ BASIC

1.3.1. Завдання значень зміним

1.3.1.1. Оператор присвоювання

Задати значення змінної можна за допомогою оператора присвоювання. Але цей найбільш простий спосіб має два істотних недоліки:

- при великій кількості вихідних даних необхідно записати в програмі відповідну кількість операторів присвоєння;
- для вирішення завдання з іншими вихідними даними потрібно змінити всі оператори присвоювання.

Формат оператора присвоювання:
[<*Рядок/мітка* >**]** <*ім'я_змінної*> = <*вираз*>

Цей оператор присвоює задане (обчислене) в правій частині значення змінній, записаній в лівій частині.

Рядок/мітка - ціле число від 1 до 32767(номер рядка) або набір символів, що починається з літери, може мати до 40 літер і/або цифр і повинен закінчуватися двокрапкою (мітка). Позначаються тільки рядки, до яких у програмі передбачена передача управління, інші позначати не потрібно.

Приклад. **A1 = 412. D=B*B-4*A*C X(I,J)=X(I,J)/X2**
Z$="Привіт" I=I+1 : Z=SIN(I+X) : D=LOG(A/X)

У правій і лівій частинах повинні бути дані одного типу (числового і числового, символьного і символьного, логічного і логічного).

До моменту виконання оператора значення змінних у правій частині повинні бути визначені.

Якщо права частина - арифметичний вираз, то в ньому можуть вживатися величини різної точності; при цьому числа однієї точності перетворяться в числа іншої точності за такими правилами:

1) значення, що присвоюють змінній, перетворяться до точності цієї змінної: **A%=23.42 [A%]=23;**

2) при перетворенні до меншої точності відбувається округлення числа:

C=55.8834567# [C]=55.88346
A%=2.5 [A%]=3

3) перетворення числа до більшої точності може змінити його зовнішнє подання

A=2.04 B#=A [B#]=2.039999961853027;

4) при обчисленні виразів операнди двомісцевих арифметичних операцій перетворяться до точності операнда з більш високою точністю, а результат - до точності змінної, якій він присвоюється.

Оператор присвоювання не має властивості алгебраїчної рівності, хоча записується подібно.

Приклад. Запис **X=X+D** в алгебрі не має сенсу. У програмуванні це означає, що до значення **X** додається значення **D** і отримане значення записується в ту ж саму комірку **X.**

1.3.1.2. Оператори DATA, READ, RESTORE

Ці оператори використовуються тоді, коли в програмі багато вхідних даних і деякі з них необхідно часто змінювати при розрахунку різних варіантів (наприклад, у процесі відлагодження програми).

Формат:

READ <*список змінних*>
DATA <*список констант*>
RESTORE [<*n*>]
<*список змінних*>=<*змінна 1*>, <*змінна 2*>,..., <*змінна N*>
<*список констант*>=<*константа1*>, <*константа2*> ,.., <*константа N*>

Значення змінних, що задаються оператором **READ**, розміщуються в списку констант оператора **DATA**, і при цьому формується спеціально відведена область пам'яті - блок даних.

Блок даних можна уявити як одновимірний масив, розміщений в оперативній пам'яті. Як тільки в програмі зустрічається оператор **DATA**, то всі константи з відповідного <*списку констант*> переносяться до блоку даних. Порядок їхнього розміщення в блоці даних у точності повторює їх порядок у списку констант оператора **DATA**. Константи можуть бути величинами будь-яких типів, і при цьому в одному списку можуть зустрічатися дані різних типів.

У програмі може бути кілька операторів **DATA**. Тоді наступна послідовність заноситься в блок даних слідом за попередньою.

Оператор **DATA** тільки зберігає дані, він є невиконуваним і може знаходитись в будь-якому місці програми. У нумерованому рядку він повинний бути єдиним.

Розміщати всі оператори **DATA** рекомендується в одному місці програми - на початку або наприкінці - це полегшує її відлагодження.

Символьні константи записуються в лапках або без них.

Оператор **READ** призначений для читання значень із блоку даних у процесі виконання програми. Він послідовно "витягає" чергове значення із блоку даних і присвоює його відповідній змінній.

READ може бути записаний у будь-якому місці програми, в будь-якому рядку з іншими операторами.

READ і **DATA** у програмі не обов'язково повинні бути парними: можуть бути присутні декілька **DATA** і один **READ** і навпаки.

Необхідно дотримуватись таких правил:

- типи змінних в **READ** повинні бути сумісні з типами значень, що присвоюють їм;
- загальна кількість змінних у всіх операторах **READ** не повинна перевищувати сумарного числа значень в операторах **DATA**.

Блок даних, за необхідності, можна використати повторно.

Оператор **RESTORE** (відновлення блоку даних) дозволяє операторові **READ** ще раз прочитати значення в зазначеному операторі **DATA**.

n - мітка або номер рядка оператора **DATA**. Без операнда **n** **RESTORE** переводить умовний покажчик на елемент блоку даних у початковий стан. При цьому стають доступними значення першого оператора **DATA**. Якщо операнд **n** вказаний, то доступними стають дані зазначеного оператора **DATA** (з міткою **n)**.

Приклад.

```
DATA 5, "група"
READ X, X$
PRINT "Іспит складає "; X; "-а "; X$
' ПОВТОРНЕ ЧИТАННЯ
READ X, X$
PRINT "Іспит складає "; X; "-а "; X$
Результат:  Іспит складає 5-а група
            Іспит складає 5-а група
```

1.3.2. Організація введення-виведення в QBasic-програмах

1.3.2.1. Оператор введення INPUT

Його використовують для зчитування вихідних даних з клавіатури.

Формат оператора:

INPUT ["<*текстовий вираз*>"**;/,**] <*список_введення*>

де < *текстовий вираз* > або підказка - необов'язковий рядок, що при необхідності відображається на екрані перед введенням користувачем даних. Він використовується для виведення на екран відповідних повідомлень про те, яку інформацію треба вводити.

Крапка з комою (**;**) після запрошення, додає знак "?" у рядок запрошення; кома (**,**) - скасовує його.

<*список_введення*> - одна або декілька змінних, розділених комами, у яких зберігаються дані, введені з клавіатури.

Якщо даних декілька, вони вводяться через "," у відповідності до списку змінних. Змінні набувають введених значень після натискання клавіші ENTER, до цього їх можна редагувати у випадку помилки, а інакше - вводити заново.

Число даних, що вводять, повинне відповідати числу змінних у списку, інакше видається повідомлення про помилку ("повторить ввод" і т.п.)

Приклад. Необхідно, щоб
A=4.5 B==-3.4E7 C$=березень

INPUT " Введіть A, B, C$ ", A, B, C$
Введіть A, B, C$ 4.5,3.4E7, березень

INPUT " Введіть A, B ", A, B
INPUT " Введіть C$ " C$
Введіть A, B, C$ 4.5,3.4E7,березень
Введіть C$ березень

1.3.2.2. Оператор виведення PRINT

Оператор **PRINT** використовується для виведення інформації на екран у стандартному форматі. Для виведення інформації до друку використовується оператор **LPRINT**.

Формат оператора:

PRINT *[*<*список_виведення*>]

LPRINT [<*список_виведення*>]

<список виведення> = <вираз1>[; /,] [< вираз2>; /,]...[< вираз n>;/,]

де *вираз* - вираз будь-якого типу, що складає *список_виведення:*

- якщо вирази у списку розділені комою (**,**) - виведення здійснюється в зонному форматі, при якому рядок умовно розділяється на 5 зон по 14 позицій у кожній і одну зону з 10 позицій. Кожне виведене значення розташовується у своїй зоні. Якщо остання зона в рядку заповнена, то виведення продовжується з першої зони наступного рядка. При виведенні чисел 1-я позиція кожної зони відводиться під знак числа. Дві коми підряд означають пропуск зони.
- якщо вирази розділені крапкою з комою (**;**) - виведення здійснюється в компактному форматі (впритул). У цьому форматі

числа виводяться через 2 або 1 позицію (якщо число зі знаком "-"). Символьні значення при виведенні розташовуються підряд. У як роздільник між виразами поряд з ";" може бути пропуск. Якщо довжина значення перевищує залишок рядка, виведення починається з початку наступного рядка.

Число стовпців (довжину рядка), як і число рядків можна задати в операторі **WIDTH**.

Приклад.

```
A=1: B=2: C=-3: C$="САТУРН"
PRINT A,B,C„C$
PRINT A;B C;C$
```

2	16	29	43	57
1	2	-3		САТУРН

2	5	7	10
1	2	-3	САТУРН

Відсутність "," або ";" викликає перехід на новий рядок і навпаки їхня наявність блокує перехід.

Приклад.

```
A=1:B=2:C=3
PRINT A,B,                          PRINT A,B
PRINT  C                            PRINT  C
1              2              3              1              2
                                              3
```

Виведення можна здійснювати з довільної позиції, розміщеної праворуч поточного положення курсора шляхом використання функції **TAB**.

TAB(<*стовпець*>)

де <*стовпець*> - номер стовпця нової позиції для друку (від початку рядка).

Приклад. **PRINT TAB(3), "Аргумент"; X, TAB(25); "Функція"; Y.**

Якщо зазначено номер позиції (<*стовпець*>), більший ніж довжина рядка виведення, то відбувається перехід на новий рядок; якщо ж (<*стовпець*>) менший за позицію курсора, то повернення назад не відбувається, а виведення здійснюється в зазначеній позиції наступного рядка. Якщо номер позиції заданий арифметичним виразом, то береться ціла частина цього виразу.

Функція **SPC**(*n*) означає виведення *n* пропусків - її використовують, якщо потрібно розділити виведені елементи декількома пропусками.

Приклад. **PRINT X;SPC(6);SIN(X).**

Позиціювання курсора на екрані перед виведенням певної інформації можна також виконати за допомогою оператора **LOCATE.**

Формат оператора:

LOCATE <*рядок*>, <*стовпець*>

де <*рядок*> і <*стовпець*> - номер рядка і стовпця, куди переміщується курсор.

Приклад. **LOCATE 15, 20: PRINT "Таблиця".**

Часто виникає необхідність оформити результати обчислень у вигляді таблиць, розташувавши їх у певних позиціях, опустити зайві знаки, додати якісь символи і т.п. В цьому випадку використовують оператор форматного виведення PRINT USING.

1.3.3. Програмування задач із використанням операторів передачі управління

1.3.3.1. Оператори умовного переходу

Умовні оператори QBasic здійснюють "розгалуження " програми, тобто передають управління на ту або іншу "гілку" обчислювального процесу залежно від результату виконання умови.

Оператор

IF...THEN...ELSE (якщо...то...інакше)

можна записати в лінійній або блоковій формі.

а) **лінійна форма** оператора IF

Якщо перевіряється одна або дві умови, то має сенс використати лінійну форму оператора **IF**

IF<*умова*>**THEN**<*оператори_так*>[**ELSE**<*оператори_ні*>]

умова - логічний вираз (набуває значення ХИБНІСТЬ або ІСТИНА).*Умовою* може бути навіть арифметичний вираз. У цьому випадку результат набуває значення ХИБНІСТЬ, якщо арифметичний вираз - нуль, і значення ІСТИНА - в інших випадках.

оператори_так – виконуються, якщо результат перевірки умови - ІСТИНА .

оператори_ні - виконуються, якщо результат перевірки умови - ХИБНІСТЬ.

Якщо ці оператори складаються з декількох операторів, то вони розділяються двокрапкою і повинні обов'язково розміщатися в одному програмному рядку.

На практиці використовують повну і скорочену форми запису оператора **IF**.

Повна форма запису (рис. 3) припускає наявність у запису гілки ELSE.

IF<*умова*>**THEN**<*дія_1*> **ELSE**<*дія_2*>
<*дія_3*>

Якщо значення логічного виразу ІСТИНА, то виконується *дія_1***,** а далі - *дія_3* **(**наступний рядок).

Якщо значення логічного виразу ХИБНІСТЬ, то виконується тіки *дія_2,* а далі - *дія_3* **(**наступний рядок).

Скорочена форма запису (рис. 4) є найпростішою формою. Вона зручна, коли частина **ELSE** відсутня. Тоді при значенні логічного вираз ХИБНІСТЬ виконання програми триває з наступного рядка.

IF<*умова*>**THEN**<*дія_1*>
<*дія_2*>

Якщо значення логічного виразу ІСТИНА, то виконується *дія_1,* а далі - *дія_2* **(**наступний рядок).

Якщо значення логічного виразу ХИБНІСТЬ, то виконується тільки *дія_2*.

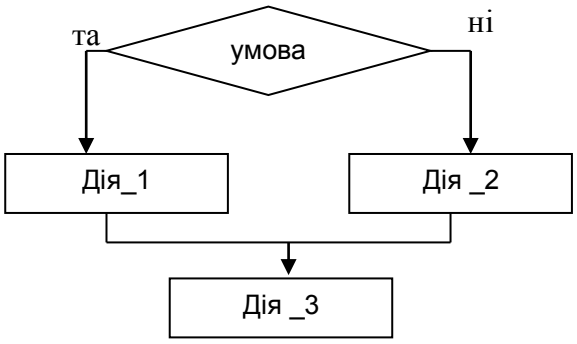


Рис. 3

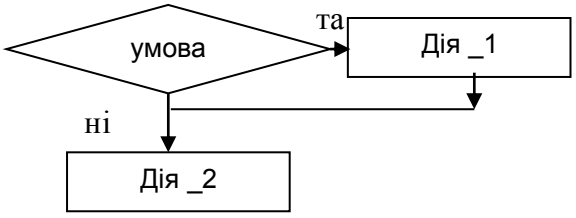


Рис.4

Приклад. Ввести числа (X,Y,Z). Визначити, чи буде найбільше з них парним. Вивести відповідне повідомлення.

```
CLS
INPUT " X ="; X:INPUT " Y ="; Y:INPUT " Z ="; Z
IF X>Y THEN MAX=X ELSE MAX=Y
IF Z>MAX THEN MAX=Z
PRINT " МАКСИМАЛЬНЕ ЧИСЛО ";MAX
IF MAX MOD 2 =0 THEN PRINT " ПАРНЕ” ELSE PRINT "НЕПАРНЕ”
```

Окремим випадком є оператор виду

IF<*умова*>**THEN [GOTO]** <*рядок*>

рядок - мітка або номер наступного виконуваного рядка.

Якщо доводиться перевіряти більше, ніж дві умови (багаторівневі перевірки), то доцільно використати вкладені оператори IF або блокову форму IF - багаторівнева побудова (більш ніж в один рядок).

Структура вкладених IF може мати такий формат:

IF<*умова1*>**THEN IF**<*умова2*> **THEN** <*оператори_так*> **ELSE** <*оператори_ні*>

<*оператори_так*> виконуються, якщо всі попередні умови мають значення ІСТИНА.

Якщо <*умова1*> - ІСТИНА, то перевіряється <*умова2*>;

Якщо <*умова1*> - ХИБНІСТЬ - управління буде передано на наступний оператор без перевірки <*умови2*>.

Якщо використовується частина **ELSE**, то вона відповідає найближчій частині **THEN**, закриваючи відповідний оператор **IF**.

Глибина вкладення обмежується тільки довжиною рядка дисплея (оператор повинний бути записаний в один рядок).

Замість вкладених операторів можна використати логічні операції - відповідне ключове слово **THEN** замінюється операцією **AND**.

Приклад.

```
IF A>B THEN IF A>C THEN PRINT "A - БІЛЬШЕ"
IF A>B AND A>C THEN PRINT "A - БІЛЬШЕ"
```

Вкладені структури з **ELSE** можуть бути досить складними, особливо якщо <*оператори_так*> містять не один, а декілька (блок) операторів. У цьому випадку використання багаторівневої структури спростить програму.

б) **блокова форма** оператора **IF**

Формат блокової форми оператора **IF**

IF <*умова1*> **THEN**
[*блок_операторів-1*]

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

умова1, умова2 - будь-які вирази, що можуть бути оцінені як ІСТИНА або ХИБНІСТЬ..

блок_ операторів-1, блок_ операторів-2,.... - один або декілька операторів в одному або декількох рядках.

Перевіряється *<умова1>*: якщо вона ІСТИНА виконуються оператори блоку **THEN**; якщо вона ХИБНІСТЬ - аналізується кожна умова **ELSEIF (**рис. 5**)**.

При виконанні будд-якої із умов буде реалізований відповідний *блок_операторів*. Якщо жодна з умов **ELSEIF** не виконується, управління передається блоку **ELSE**, а якщо його немає - операторові, розташованому за **END IF**. Блокова структура може мати будь-яку кількість умов **ELSEIF**.

Виконання кожного з альтернативних блоків автоматично завершується переходом на **END IF**.

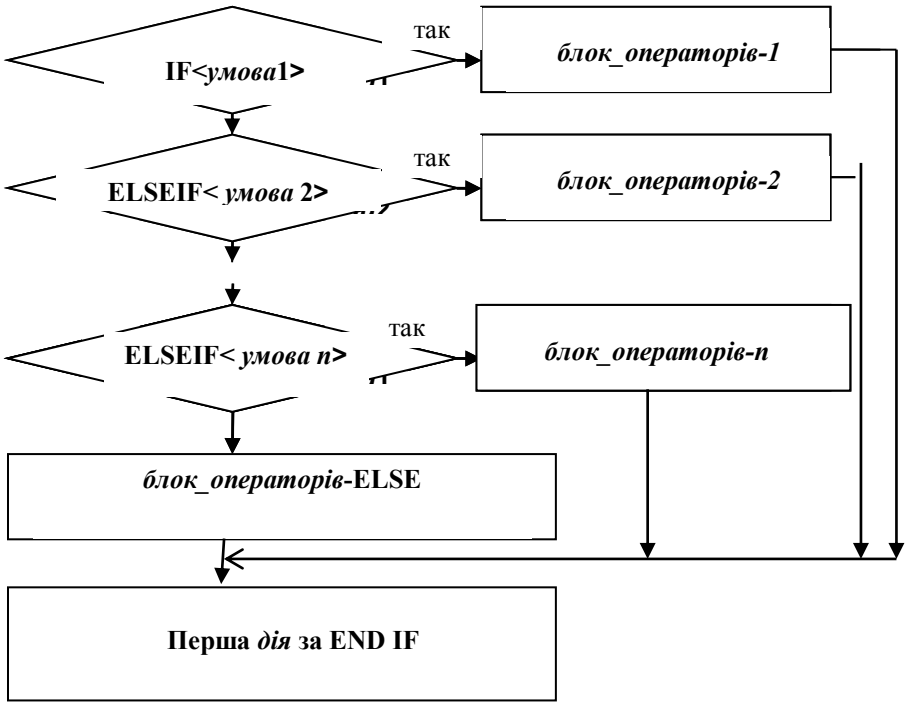


Рис. 5

Приклад. Ввести ціле число від 1 до 999. Визначити, скільки цифр воно містить. Якщо введено більше число, вивести відповідне повідомлення.

```
CLS:INPUT "Введіть число X=",X
IF X <10  THEN
        Y=1
ELSEIF X < 100 THEN
        Y=2
ELSEIF X<1000 THEN
        Y=3
```

```
ELSE
        PRINT " Введено більше число ": STOP
END If
PRINT "число  "; X ; " містить "; Y; " цифр "
```

Будь-який з *блоків_операторів* може містити вкладені блокові структури.

Оператор SELECT CASE ... END SELECT (вибрати варіант)

Цей управляючий оператор доцільно використовувати при перевірці складних умов. Він виконує один з декількох *блоків_операторів* залежно від результатів перевірки заданих умов. Формат оператора:

SELECT CASE <*вираз вибору*> (<*тест-вираз*>)
CASE <*список_виразів1*>
[*блок_операторів-1*]
[**CASE** <*список_виразів2*>
[*блок_операторів-2*]]...
[***CASE ELSE***
[*блок_операторів-n*]]
END SELECT

<*вираз вибору*> або <*тест-вираз*> - будь-який числовий або символьний вираз.

<*список_виразів*> - один або кілька виразів для порівняння з виразом вибору.

У вираз перед будь-яким знаком відношення додається спеціальне ключове слово IS

<*блок_операторів-1*>, … ,<*блок_операторів-n*> - один або кілька операторів, записаних в одному або декількох рядках.

Існують різні способи запису умовних виразів у блоках **CASE**. Аргументи списку виразів можуть набувати кожну з таких форм або їхню комбінацію і повинні розділятися комами:

значення - вираз і значення перевіряються на рівність один одному;

значення1 TO значення 2 - перевіряється, чи належить *тест-вираз* області [*значення2 – значення1*] (менше значення повинне бути першим). Якщо належить, то виконується відповідний блок операторів;

IS *знак_відношення вираз*

вираз - будь-який числовий або символьний вираз, сумісний з виразом вибору.

знак_відношення - один із знаків відношення.

Якщо *вираз_ вибору* потрапляє в діапазон, зазначений в **CASE**, програма виконує відповідний *блок_операторів*.

Умова, що перевіряється, може мати і більш складний вид:

CASE IS<0, IS>90

Якщо *вираз_вибору* відповідає умовам *списку_виразів* даного блоку **CASE**, то виконуються оператори цього блоку. Якщо жодна з умов не виконується, управління передається **CASE ELSE**; якщо **CASE ELSE** немає - операторові, що розташований за **END SELEC**T. Якщо *вираз_вибору* задовольняє декільком умовам **CASE**, виконується *блок_операторів*, що йде першим.

Блоки **SELECT CASE** можуть бути вкладеними. Кожний блок повинен завершуватися **END SELECT**.

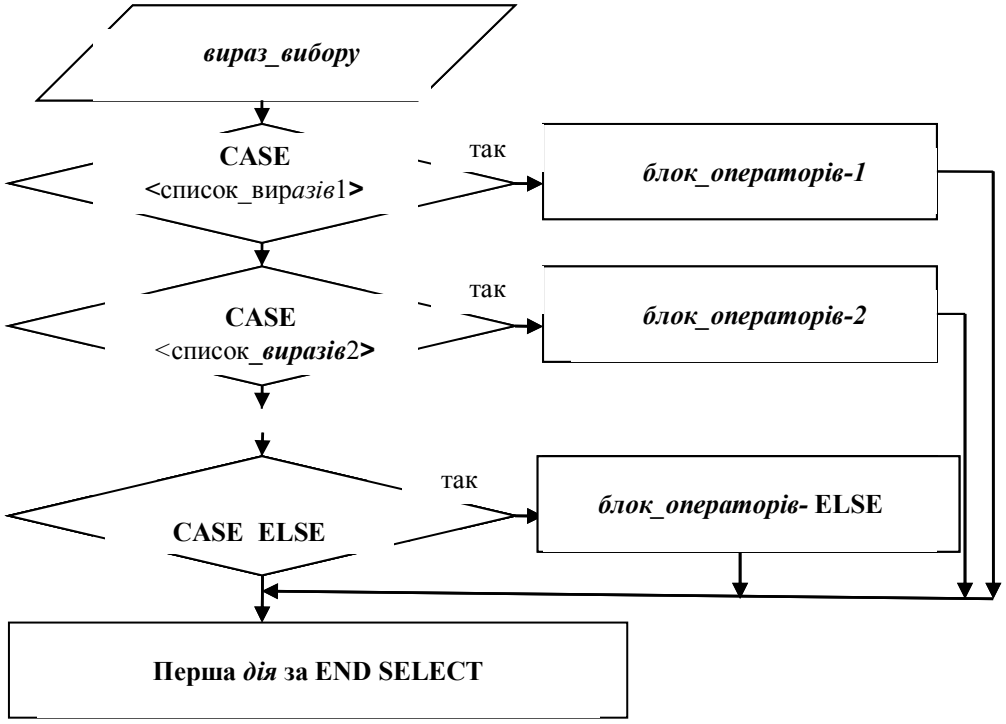


Рис.6

Приклад. Скласти програму, яка підраховує кількість цифр введеному числі

```
CLS:INPUT "Введіть число X=",X
SELECT CASE X
CASE 1,2,3,4,5,6,7,8,9
 PRINT “Одна цифра”
CASE 10 TO 99
 PRINT “Дві цифри”
CASE IS>999
 PRINT “Введено більше число”
CASE ELSE
PRINT “Три цифри”
END SELECT
```

Приклад. Скласти програму, яка за введеним номером місяця виводить на екран назву пори року.

```
 CLS:  INPUT " Введіть номер місяця";K
SELECT CASE K
CASE 1, 2, 12
 PRINT “Зима”
CASE 3, 4, 5
PRINT “Весна”
CASE 6 TO  8
PRINT “Літо”
CASE IS>9, IS<11
PRINT “Осінь”
CASE ELSE
PRINT " Такого місяця немає "
END SELECT
```

1.3.3.2. Оператори безумовного переходу

В QBasic є можливість нумерації програмних рядків і звертання до них за номером.

[<*номер рядка/мітка* >] **GOTO** <*номер рядка/мітка* >

виконує безумовний перехід до вказаного рядка.

Цей оператор повинен бути або єдиним у рядку, або останнім у багатооператорному рядку.

Приклад.

```
GOTO 67
8 GOTO MP
....
MP: A=5
S=0:N=N+1:GOTO 120
```

Ще одним різновидом оператора безумовного переходу є оператор множинного вибору.

Формат оператора:

ON <*показни_ переходу*> **GOTO** <*номер рядка/мітка* >

виконує перехід до однієї із дій, позначених мітками залежно від значення виразу, записаного в *показнику_переходу*. Вираз набуває значення в діапазоні від 0 до 255.

При реалізації оператора обчислюється ціла частина цього виразу (при потребі з округленням). Якщо вона дорівнює 1, то управління передається на першу мітку (на відповідний рядок); якщо - 2, то на другу і т.д. Якщо ціла частина =0 або > числа міток, то управління передається операторові, що розташований за ON...GOTO; якщо ж ціла частина < 0, то буде повідомлення про помилку.

Приклад.

```
20 INPUT „Ввести номер елемента N=”, N
   ON N GOTO 100,150,80,210
   GOTO 999
80 PRINT "ЛІТІЙ":       GOTO 20
100 PRINT "КИСЕНЬ": GOTO 20
150 PRINT "ГЕЛІЙ":     GOTO 20
210 PRINT "БАРІЙ":      GOTO 20
999 END
```

1.3.3.3. Оператори циклів

Призначені для реалізації циклічних процесів.

Оператор циклу складається із двох частин. Перша частина - це заголовок циклу, друга частина - кінець циклу. Між ними розміщається задана послідовність операторів - тіло циклу.

Цикли можна реалізувати за допомогою умовних операторів, але це досить незручно.

Операторами циклів QBasic є:

FOR... NEXT
WHILE... WEND
DO ... LOOP

Оператор **FOR ... NEXT**

Цей оператор дозволяє організувати виконання блоку операторів певну кількість разів, яка задається безпосередньо в операторі або може бути легко обчислена до початку виконання.

За типом - це цикл з передумовою.

Формат оператора:

FOR <*параметр циклу*> =<*початок*> **TO** <*кінець*>[**STEP** <*збільшення*>**]**
[*тіло циклу*]
NEXT [*параметр циклу*]

FOR - визначає початок циклу (для); **TO** - визначає кінець циклу (до); **STEP** - крок зміни параметру циклу; **NEXT** - кінець (наступний)

- *параметр циклу* - числова змінна, яка визначає число повторень циклу;
- *початок, кінець* - арифметичні вирази, які визначають початкове і кінцеве значення параметру циклу;
- *збільшення* (крок) - числова константа або арифметичний вираз, що визначає зміну параметру циклу при кожному кроці циклу. Якщо *збільшення* не задано, то за замовчуванням воно приймається рівним одиниці;

• *тіло циклу* (блок операторів) - набір операторів, що виконуються багаторазово.

У процесі виконання цього оператора реалізуються такі дії:

1)обчислюються значення арифметичних виразів (*початок, кінець, збільшення*).

2)*параметру циклу* присвоюється початкове значення (формується лічильник циклу).

3)перевіряється умова: якщо *збільшення* >0 і *параметр* ≤ кінцевого значення або *збільшення* <0 і *параметр* ≥ кінцевого значення, то виконується *тіло циклу*; значення *параметра* змінюється на величину *збільшення* і повторюється п.3. У протилежному випадку виконуються програмні рядки, що розташовані за оператором **FOR ... NEXT**.

При *збільшенні* = 0 цикл стає нескінченним.

Правила застосування оператора **FOR ... NEXT:**

• параметр циклу, зазначений в **FOR**, повинен збігатися з параметром, зазначеним в **NEXT**;

• не можна передавати управління у середину циклу - у цьому випадку *параметр* циклу не визначений;

• *параметр* циклу, *початок*, *кінець*, *збільшення* не можна змінювати у середині циклу;

• можна виходити із циклу природно (тобто після виконання його задане число разів), а також за допомогою управляючих операторів з будь-якого оператора тіла циклу.

Можна також вийти із циклу, не дочекавшись виконання всіх повторень, скориставшись альтернативним виходом із циклу за допомогою оператора **EXIT FOR**. Управління буде передано операторові, що розташований після **NEXT**.

• після завершення циклу значення *параметру* дорівнює кінцевому значенню плюс *збільшення*;

• із циклу припустиме звертання до підпрограми з наступним поверненням у нього.

Приклад. Програма обчислення суми натурального ряду чисел від 1 до N

```
CLS: INPUT "  ВВЕДІТЬ ЗНАЧЕННЯ N=";N
S=0
FOR A=1 TO N STEP 1
    S=S+A
NEXT A
PRINT " СУМА N ЧИСЕЛ НАТУРАЛЬНОГО РЯДУ= ";S
```

Цикли **FOR ... NEXT** можуть бути вкладеними. Кожний вкладений цикл повинен мати свої ідентифікатори параметру.

Оператор **NEXT** для внутрішнього циклу повинен виконуватися раніше оператора **NEXT** для зовнішнього циклу. Глибина вкладення обмежується тільки розміром доступної пам'яті в робочій області QBasic.

Якщо кілька циклів мають одну загальну кінцеву точку, то можна вказати для них один оператор **NEXT**, перелічивши в ньому параметри циклів у порядку, зворотному їхній вкладеності (від внутрішніх - до зовнішнього).

Приклад.

```
FOR A=1 TO 3
        FOR B=5 TO 14
                C=A+B
                PRINT C
        NEXT B                  NEXT B, A
NEXT A
```

Оператор **DO ... LOOP**

Найбільші можливості (зокрема виконання тіла циклу з умовою закінчення як на початку, так і наприкінці циклу) надає конструкція, що відповідає циклу типу **DO ... LOOP**.

Оператор має 4 форми запису:

Перевірка угорі (цикл з передумовою - "ПОКИ")

DO [{**WHILE**|**UNTIL**}<*умова*>]
[*тіло циклу*]
LOOP,

де **DO** - виконати, **WHILE** - поки, **UNTIL** - поки не, **LOOP** – цикл; *умова* - вираз, що QBasic оцінює як ІСТИНА або ХИБНІСТЬ.

Повторюється тіло циклу, поки *умова* вірна (**WHILE**) або поки *умова* не стане вірна (**UNTIL**).

Умова перевіряється на початку циклу, тому *тіло циклу* може взагалі жодного разу не виконатися, коли результат першої перевірки – ХИБНІСТЬ (рис. 7) або ІСТИНА (рис.8).

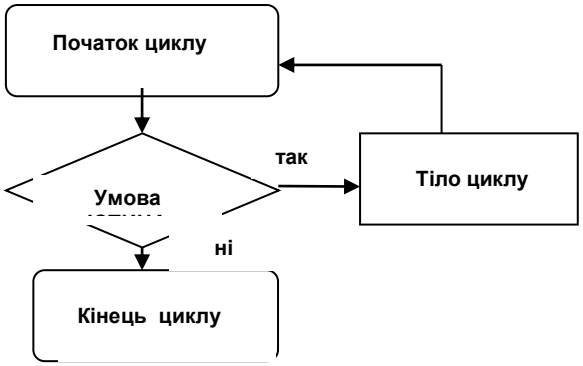


Рис. 7 DO UNTIL

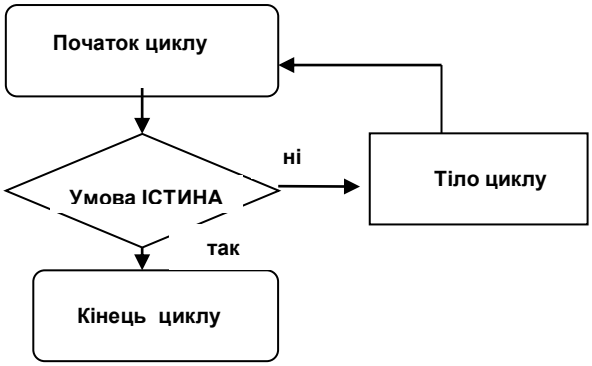


Рис. 8 DO UNTIL

Перевірка внизу (цикл із післяумовою -"ДО")

DO
[*тіло циклу*]
LOOP[{**WHILE**|**UNTIL**}<*умова*>]

Перевірка умови виконується наприкінці циклу, у зв'язку із цим тіло циклу завжди виконується хоча б один раз (рис. 9, 10).

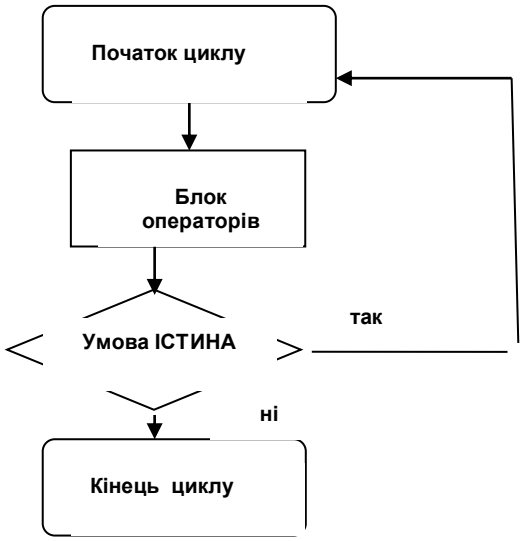


Рис. 9 DO LOOP WHILE

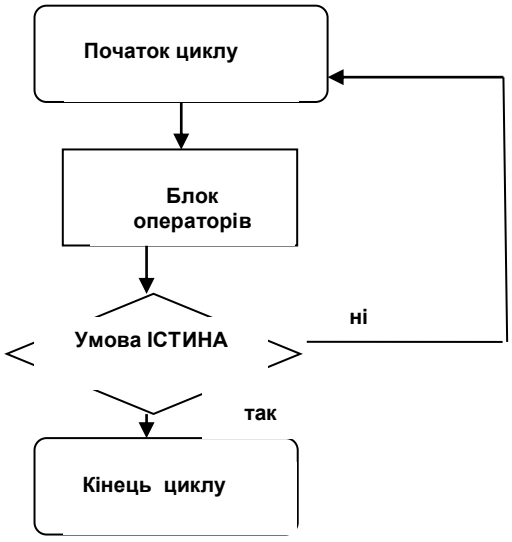


Рис. 10 DO LOOP UNTIL

DO у спрощеному вигляді

DO
[*тіло циклу]*
LOOP

створює нескінченний цикл (з нього можна вийти, використовуючи оператор **EXIT DO** або інший управляючий оператор).

Наведемо приклади реалізації задачі обчислення суми натурального ряду чисел за допомогою цього оператора

Приклад. Цикл з передумовою

```
CLS: INPUT "  ВВЕДІТЬ ЗНАЧЕННЯ N=";N
    S=0
    A=1
    DO WHILE A<= N
        S=S+A
        A=A+1
    LOOP
    PRINT " СУМА N ЧИСЕЛ НАТУРАЛЬНОГО РЯДУ= ";S
```

Приклад. Цикл з післяумовою

```
    CLS: INPUT "  ВВЕДІТЬ ЗНАЧЕННЯ N=";N
    S=0
```

```
A=1
DO
    S=S+A
    A=A+1
LOOP UNTIL A>N
PRINT " СУМА N ЧИСЕЛ НАТУРАЛЬНОГО РЯДУ= ";S
```

1.3.4. Деякі корисні команди Basic

Розглянемо деякі команди QBasic, що часто використовуються при написанні програм.

Команда **CLS**. Очищує з екрана монітора отримані раніше результати. Має такий вид:

CLS

SLEEP[*n*] (Пауза). Для організації паузи при роботі програми. Вона затримує виконання програми на *n* секунд. Крім того, для цього можна використати команду введення.

Приклад. INPUT P$, де P$ - деяка змінна.

Якщо паузу потрібно закінчити, то варто натиснути алфавітно-цифрову клавішу і ENTER.

Команду-коментар **REM** використовують для внесення пояснень у текст програми. **REM** можна замінити апострофом - **'**).

REM <*текст пояснення*>

або

' <*текст пояснення*>

Ця команда може знаходитися в будь-якому місці програми, але в рядку повинна бути єдиною або останньою. Вона не впливає на виконання програми.

Команда **STOP**. Команда зупинки виконання програми може знаходитись в будь-якому місці програми. Часто її використовують, щоб переписати проміжні результати з екрана монітора. Виконання програми можна продовжити за допомогою натискання клавіші F5.

Команда **END** - команда закінчення роботи програми.

Команда **SWAP**. Дозволяє виконати обмін значеннями між двома змінними. Команда має вигляд **SWAP** A, B, де A и B - імена змінних, які обмінюються значеннями.

Приклад. У результаті виконання програми

```
A=7:B=5
SWAP A,B
PRINT A,B
```

на екран монітора будуть виведені значення 5 і 7.

У класичному Basic для обміну даними необхідно вводити проміжну змінну, і програма може виглядати так:

```
A=7: B=5
C=A: A=B: B=C
PRINT A,B
```

1.3.5. Програмування задач обробки масивів даних

У практиці інженерних розрахунків розв'язання багатьох задач припускає обробку великих сукупностей однорідних даних, **наприклад**, множина результатів вимірів однієї й тієї ж фізичної величини. Якщо всі елементи об'єкта належать до однакового типу, то змінна, якою позначені ці елементи, є однорідною і може бути подана у вигляді деякого масиву.

Масив - сукупність однорідних елементів, упорядкована за допомогою числових значень індексів.

Індекс – константа, змінна або вираз. Значення індексу повинне бути цілим додатним числом (бо це номер), тому він завжди округляється, перетворюється і зберігається в такому вигляді.

Ім'я масиву або **ідентифікатор** позначає всю впорядковану множину елементів у цілому.

Елементом масиву називають змінну з індексом. Для позначення окремого елемента масиву до імені додається список індексів, що дозволяє здійснювати доступ до конкретного елементу.

Список індексів - упорядкована послідовність індексів, розділених комами. Кожний індекс має свій діапазон зміни, названий **граничною парою**.

Змінні з індексом обробляються за такими ж правилами, як і скалярні (прості) змінні.

Всі елементи одного масиву не тільки мають спільне ім'я, але й розміщаються, як правило, у послідовних комірках пам'яті ЕОМ.

Місце розташування елемента може визначатися як одним (одновимірні масиви), так і декількома значеннями (багатовимірні масиви) індексів.

Характеристиками масивів є вимірність та розмір.

Кількість індексів визначає **вимірність** масиву.

Кількість елементів масиву визначає **розмір** масиву.

В алгоритмах обробки масивів наявні блоки обробки елементів масивів і величин відповідних індексів. Як правило, ці блоки є складовими циклічних алгоритмів. Це пов'язане з тим, що обробка будь-якого елемента масиву являє собою однакову

послідовність дій і, організувавши цикл по числу елементів масиву, можна тими самими командами обробити всі елементи.

Особливість таких алгоритмів - при кожному повторенні циклу в обробці повинен брати участь наступний елемент. Тому їх називають **циклами з переадресацією** - перехід до наступного елемента здійснюється збільшенням адреси звертання до пам'яті (як правило, на 1).

Відмітимо, що обробляти увесь масив у цілому засобами QBasic не можливо.

Максимальне значення індексу 32767, мінімальне (за замовчуванням) - 0. При необхідності за допомогою оператора **OPTION BASE** *N* (*N* =0/1) воно може бути встановлено рівним 1. Надалі у прикладах вважатимемо, що *N* дорівнює 1.

Масиви, що використовуються в програмах, повинні бути обов'язково в них описані (оголошені).

Для опису масиву в QBasic використовується вже згадуваний в п.1.2.4.2 оператор-описувач **DIM**, що розташовується в програмі раніше, ніж починається робота з відповідним масивом. Як правило, ці оператори розташовують на початку програми для полегшення відлагодження.

Застосувавши оператор **DIM** ви одночасно:

- визначаєте ім'я масиву;
- описуєте тип елементів масиву або вказуєте тип даних для скалярних змінних (не масивів);
- обнуляєте всі елементи числових масивів, а символьним - присвоюєте значення порожнього рядка (" ");
- резервуєте комірки пам'яті для елементів масивів.

Формат оператора:

DIM[**SHARED**]<*змінна*>[(<*індекс*>)][**AS** *тип*][,<*змінна*> [(<*індекс*>) [**AS** тип]]

- **SHARED** вказує, що змінні використовуються спільно всіма процедурами **SUB** або **FUNCTION** у модулі.
- <*змінна*> - ім'я змінної масиву.
- <*індекс*>-розміри масиву у вигляді:[*низ* **TO**] *верх*[, [*низ* **TO**] *верх*]...
- *низ* - нижня границя індексу масиву. За замовчуванням нижня границя дорівнює нулю; *верх* - верхня границя.
- **AS** *тип* - описує тип елементів масиву або скалярну змінну (INTEGER, LONG, SINGLE, DOUBLE, STRING або тип даних , що задається користувачем).

Приклад.

DIM RUB(10) - одновимірний масив з 10 речовинних елементів.
DIM MBA%(3,4) - двовимірний масив з 12 цілочислових елементів (матриця розміром 3*4).
DIM All_Bright$(5) - одновимірний масив з 5 символьних елементів.
DIM X(88), XX(89), XXX(8,9) - два одновимірних і один двовимірний масиви.
DIM SOS (3 To 12) - двовимірний масив з 10 речовинних елементів (індексами змінюються від 3 до 12).
DIM W#(3 To 10, 0 To 5) - двовимірний масив з 48 речовинних довгих елементів.
DIM Y45 (1 To 45) AS INTERGER - одновимірний масив з 45 цілочислових елементів.

Приклади програмування задач обробки одновимірних та двовимірних масивів

Задача 1

З елементів числового масиву A(20) створити два масиви X і Z: масив X - з від'ємних елементів масиву A, масив Z - з елементів із ключовою ознакою A(i) >12.

```
DIM A(20),X(20),Z(20)
FOR I = 1 TO 20
   PRINT " ВВЕДІТЬ A(";I;")=";
   INPUT A(I)
NEXT I
J=0: K=0
FOR I = 1 TO 20
   IF A(I)<0 THEN
        J=J+1
        X(J)=A(I)
        PRINT "X(";J;")=";X(J)
   ELSEIF A(I)>12 THEN
        K=K+1
        Z(K)=A(I)
        PRINT "Z(";K;")=";Z(K)
   END IF
NEXT I
```

Задача 2

Заданий двовимірний числовий масив X(k, k) (k рядків і k стовпців).

Визначити різницю R між добутком і сумою елементів масиву, які задовольняють умові X(i, j) > 10;

визначити середні арифметичні значення елементів у кожному рядку окремо і записати їх в одновимірний масив Y(k);

визначити кількість від'ємних елементів у кожному стовпці масиву X(k, k), результат помістити в масив Z(k);

визначити значення максимального елемента на головній діагоналі масиву X(k, k)

```
CLS: INPUT “ВВЕДІТЬ ЧИСЛО РЯДКІВ (СТОВПЦІВ) ”;K
DIM X(K,K),Y(K),Z(K)
FOR I= 1 TO K
        FOR J = 1 TO K
                PRINT “ВВЕДІТЬ X(”;I; J; “)=”;
                INPUT X(I,J)
        NEXT J
NEXT I
S=0:P=1
FOR I=1 TO K
        FOR J = 1 TO K
                IF X(I,J)>10 THEN S=S+X(I,J):P=P*X(I,J)
        NEXT J
NEXT I
R=P-S
PRINT “R=”;R
FOR I = 1 TO K
        Y(I)=0
        FOR J = 1 TO K
                Y(I)=Y(I)+X(I,J)
        NEXT J
        Y(I)=Y(I)/K
        PRINT “Y(“;I;”)=”;Y(I)
NEXT I
FOR J = 1 TO K
        Z(J)=0
        FOR I = 1 TO K
                IF X(I,J)>0 THEN Z(J)=Z(J)+1
        NEXT I
        PRINT “Z(”;J; “)=”;Z(J)
NEXT J
MAX=X(1,1)
FOR I = 1 TO K
        IF X(I,I)>MAX THEN MAX=X(I,I)
NEXT I
PRINT “MAX=”;MAX
```

1.4. РОЗРОБЛЕННЯ ПРОГРАМ З ВИКОРИСТАННЯМ ТАБЛИЧНИХ ФОРМ

Результати експериментальних досліджень, статистичні дані доцільно подавати у зручному для сприйняття вигляді таблиць.

Формат виведення інформації за допомогою оператора **PRINT** не дозволяє одержувати якісні документи через те, що відображувані значення можуть мати різну кількість розрядів і при цьому дані виявляються зміщеними, що приводить до скривлення стовпчиків таблиці.

Необхідний нестандартний формат виведення, що дозволяє самостійно вирішувати, у якому місці виведеного рядка помістити те або інше числове значення, скільки позицій варто відвести цілій і дробовій частинам і т.д.

Для цього призначений оператор **PRINT USING.**
Формат оператора:

PRINT USING "<*список_форматів*>"; <*список_ виведення* >,

де <*список_форматів*>- один або кілька *визначників_формату* (форматних полів); <*визначник_формату*> - текстовий вираз (у подвійних лапках), що складається із символів управління форматом виведення. Символи бувають звичайні й спеціальні.

<*список__ виведення* >- числові або символьні вирази, розділені крапкою з комою (проміжок і кома автоматично перетворяться в крапку з комою)

Виведення чисел

Спеціальні символи для виведення числових даних:

#	^	$	.(крапка)	, (кома)

- символ # резервує цифрову позицію. Група # задає числове поле виведення;
- крапка в описі числового поля задає позицію десяткової крапки, що відокремлює цілу частину від дробової. Число вирівнюється по правому краю, а зайві позиції ліворуч заповнюються проміжками. При необхідності виконується округлення *(не усікання*). Зайві позиції після крапки заповнюються нулями. Якщо задане числове поле мале (не вміщується ціла частина або знак числа), то виводиться символ **%**, а після нього - повне значення числа;

Приклад.

PRINT USING "###.##"; 524.458 **результат: 524.46**
PRINT USING "##.###"; -7.14 **результат: -7.140**
PRINT USING "R=##.#"; 16.445 **результат: R=16.4**
PRINT USING "###.##"; 72445.66 **результат: %72445.66**
PRINT USING "######"; 333333 **результат: 333333**
PRINT USING "#.##"; 9.99745 **результат: %10.00**

- чотири підряд знаки піднесення степеня визначають місцезнаходження символу E або D, знака порядку і порядку числа, тобто число виводиться в плаваючому форматі.

Приклад.

PRINT USING"##.###^^^^";433.555 **результат:4.336E+02.**

- сполучення символів $$, що передує шаблону числового поля, приведе до появи знака грошової одиниці $ перед старшою значущою цифрою числа.

Приклад.

PRINT USING "$$###.##"; 678.910 результат: $678.91
PRINT USING "$###.##"; 678.910 результат: $ 678.91

- кома у форматному полі використовується для відділення у великих числах тисяч і мільйонів

Приклад.

PRINT USING"$$###,###.##";2115.5 результат:$2,115.50

Виведення текстів

Спеціальні символи для виведення текстових даних:

- **&** - виводить текстовий вираз цілком.

Приклад .

PRINT USING "&"; "КАЛИГУЛА"
результат: КАЛИГУЛА

- **\ n** проміжків \ - виводить (n+2) початкових символів текстового виразу.

Приклад.

PRINT USING "\ب ب ب\";" КАЛИГУЛА "
результат: КАЛИГ

- **!** - виводить тільки перший символ текстового виразу.

Приклад.

PRINT USING "! "; " КАЛИГУЛА "
результат: К

- якщо для оформлення результатів необхідно включити спеціальні символи, то перед будь-яким спеціальним з них у шаблоні розміщають знак “підкреслення” ("_").

Приклад.

PRINT USING "Вагони _### _& _###"; 10; 12
результат: Вагони #10 & #12

- якщо *список форматів* містить кілька визначників формату, а *список виведення* - кілька елементів, то перший елемент *списку виведення* виводиться по першому формату, другий - по другому і т.д. Елементи виводяться без пропусків, тому числові формати рекомендується розділяти символьними константами.

Приклад.

PRINT USING "X=#.# F=####";.151,366.3
результат:X=0.2 F=366

Коли *список форматів* буде вичерпаний, *визначники формату* почнуть вибиратися з початку списку.

Приклад.

PRINT USING "##.##";10.1,-4.828
результат:10.10 -4.83

Можна оформити *список форматів* у вигляді символьної змінної (форматного рядка), а в операторі зробити посилання на цей рядок.

Приклад.

```
adr$ = "Дім ##, Квартира ##"   d = 48: k = 97
PRINT USING adr$; d; k
результат: Дім 48 Квартира 97
```

Для зображення таблиць можливе використання символів, наведених на клавіатурі, а також символів псевдографіки: "-" "=" "," "|" ┌ ╗ ╤ і т.ін.

Приклад. Програмування задачі з виведенням результатів у вигляді табличної форми.

Постановка задачі

Задано інформацію про успішність 10 студентів групи.
Атрибути запису:
прізвище - до 15 символів, ім'я - до 10 , по батькові - до 10,
рік народження - 4, стать - до 5, оцінка з математики - 3, оцінка з фізики - 3, оцінка з хімії - 3
Необхідно:
1) вивести вихідні дані у вигляді таблиці:

Підсумки сесії групи 5-ІІІ-В

№	Прізвище	Ім'я	По батькові	Стать	Рік	Мат	Фіз	Хім
1	Токарєв	Іван	Борисович	ч	1983	4	2	5
2	Рудь	Ганна	Сидорівна	ж	1984	4	3	4
…	…...	…...	…...	...	…...	…...	…...	…...

2) вивести дані про наймолодшого студента:
НАЙМОЛОДШИЙ СТУДЕНТ
Рудь Г.С. 1984р.

3) вивести інформацію про боржників, якщо їх немає - видати відповідне повідомлення:
Боржник: Токарєв Іван Борисович
оцінка з математики - 4
оцінка з фізики - 2
оцінка з хімії - 5
або
Боржників немає

4) підрахувати й вивести відсоткове співвідношення чоловіків і жінок у групі
У групі 5-ІІІ-В - 50% чоловіків і 50% жінок

Складемо таблицю ідентифікаторів (табл. 4).

Таблиця 4

Ідентифікатори

№ п/п	Іденти-фікатор	Опис (тип ідентифікатора)
1	N	Кількість записів у таблиці (числ.)
2	I	Поточний номер студента в списку (числ.)
3	max	Найбільший рік народження (числ.)
4	imax	Номер у списку студента з найбільшим роком народження (числ.)
5	PR	Ознака наявності боржників у групі (1- боржники є, 0 - немає)(числ.)
6	CM,CG	Число чоловіків і жінок (числ.)
7	PM,PG	Відсоток чоловіків і жінок (числ.)
8	NG$	Найменування групи (симв.)
9	L$	Горизонтальна лінія (симв.)
10	FORM$	Формат для виведення рядка таблиці (симв.)
11	FAM$(N)	Масив прізвищ (симв.)
12	IM$(N)	Масив імен (симв.)
13	OT$(N)	Масив по батькові (симв.)
14	POL$(N)	Масив ознак статі (симв.)
15	GOD(N)	Масив років народження (числ.)
16	OCM(N)	Масив оцінок з математики (числ.)
17	OCF(N)	Масив оцінок з фізики (числ.)
18	OCH(N)	Масив оцінок з хімії (числ.)

Програма

```
CLS:INPUT "Введіть кількість записів  таблиці: ", N
PRINT
DIM FAM$(N), IM$(N), OT$(N), POL$(N), GOD(N)
DIM OCM(N), OCF(N), OCH(N)
INPUT "Найменування групи: ", NG$
FOR I = 1 TO N
PRINT "Введіть"; I; "- й ЗАПИС:"
INPUT "прізвище";FAM$(I)
INPUT "ім'я";IM$(I)
INPUT "по батькові";OT$(I)
INPUT "стать ";POL$(I)
INPUT "рік народження  ";GOD(I)
INPUT "оцінка з математики ";OCM(I)
INPUT "оцінка з фізики ";OCF(I)
INPUT "оцінка з хімії ";OCH(I)
PRINT
NEXT I
L$=STRING$(54,"-")
CLS:PRINT "Підсумки сесії групи ";NG$
PRINT
PRINT L$
PRINT "| № | Прізвище | Ім'я | По батькові | Стать | Рік |Мат|Фіз|Хім|"
PRINT L$
FORM$="|###|\                \|\          \|\           \|\     \|####|###|###|###|:"
```

```
FOR I = 1 TO N
    PRINT USING FORM$ ;I; FAM$(I), IM$(I), OT$(I), POL$(I), GOD(I),
OCM(I), OCF(I),OCH(I)
NEXT I
PRINT L$
PRINT
' Визначення наймолодшого студента
MAX=GOD(1)
FOR I=1 TO N
    IF GOD(I)>=MAX THEN MAX=GOD(I):IMAX=I
NEXT I
PRINT "НАЙМОЛОДШИЙ СТУДЕНТ"
PRINT USING ="\                    \ !.!.  #### р.";FAM$(IMAX), IM$(IMAX),
OT$(IMAX), GOD(IMAX)
' визначення боржників
PR=0
FOR I=1 TO N
    IF OCM(I)<3 OR OCF(I)<3 OR OCH(I)<3 THEN
        PR=1
        PRINT "Боржники:"
PRINT FAM$(I);" "; IM$(I));" " OT$(I));" ";POL$(I);" ";GOD(IMAX);" р.";"
мат:"OCM(I);" фіз:"OCF(I);.";" хім:"OCH(I)
    PRINT
    END IF
NEXT I
IF PR=0 THEN PRINT "Боржників немає"
' Визначення відсотка чоловіків і жінок
CM=0: CG=0
FOR I=1 TO N
    IF POL$(I)="ж"  OR  POL$(I)="Ж" THEN CG=CG+1 ELSE CM=CM+1
NEXT I
PM=(CM/N)*100: PG=(CG/N)*100
PRINT "У групі ";NG$; " чоловіків ";PM; "%"; ", жінок ";PG;"%"
```

1.5. МОДУЛЬНЕ ПРОГРАМУВАННЯ

При вирішенні багатьох практичних задач виникає необхідність у багаторазовому обчисленні величин за тими ж самими алгоритмами при різних значеннях змінних.

Щоб програма була компактною, доцільно оформити ці обчислення у вигляді програмних модулів (процедур) і у відповідних місцях основної програми звертатися до них, задаючи необхідні для розрахунків значення параметрів.

Крім того, розбивання складної задачі на незалежні модулі є надзвичайно привабливою ідеєю, бо полегшує процеси створення і відлагодження програми.

Самостійно модулі не виконуються. Спочатку виконується частина програми, що називається **основною (головною)** програмою. Інші модулі є допоміжними і викликаються в процесі виконання основної програми. Після виконання чергового модуля

знову продовжує виконуватися основна програма (крім спеціальних випадків).

Модулі можуть викликати інші модулі. У програмі може бути будь-яка кількість модулів, які обмінюються значеннями не тільки з основною програмою, але й між собою.

Розглянемо, яку роль відіграють **процедури** в основній програмі, як їх створювати.

Процедури типу **subprogram** і **function** - є ефективним засобом модульного програмування для мов класу Basic.

Поряд із цим QBasic підтримує колишні засоби організації модулів - **subroutine** і **DEF FN**. Це дозволяє використовувати в QBasic програми, написані в старих версіях Basic.

1.5.1. Типи процедур. Основні визначення

Процедура - це закінчена частина програми, що призначена для вирішення певної задачі. Будь-яка досить складна програма може бути розбита на основну програму й процедури, у яких вирішуються певні підзадачі.

QBasic дозволяє створювати два типи процедур - це підпрограми (**SUB**) і функції (**FUNCTION**). Різниця між ними полягає в способі виклику з головної програми й поверненні обчислених значень у головну програму.

Властивості процедур

Процедури характеризуються трьома основними властивостями:

- вони відділені від основної програми;
- можуть оперувати зі змінними двох типів:

такими, що використовуються тільки в середині процедури (локальні змінні);

такими, що є спільними для процедури та основної програми (глобальні змінні);

- мають можливість одержувати інформацію з головної програми у вигляді параметрів або через загальну область і повертати обчислені значення назад.

Перша властивість робить положення процедури двояким: з одного боку, вона залишається частиною програми (якщо програма зберігається, процедура записується в той же файл), з іншого, - існує якби незалежно: в QBasic для створення (редагування) різних процедур використовуються різні вікна.

Друга властивість означає, що, будучи незалежною частиною програми, процедура може використати свої власні змінні - вони не залежать від змінних основної програми та інших

процедур. Так, змінні, які мають однакові імена у головній програмі та у процедурі, - це різні змінні, які існують і використовуються незалежно одна від іншої. Але є можливість зробити будь-яку змінну доступною для основної програми, яка матиме можливість змінювати її значення.

Третя властивість процедур - це можливість розв'язувати поставлену задачу з різними вихідними даними, які їм передаються з основної програми.

Під **формальними** слід розуміти параметри, які вказуються в описі процедури. Вони є локальними змінними й з їхньою допомогою можна змінювати значення змінних у процедурі.

Під **фактичними** слід розуміти параметри, які вказуються у виклику процедури. Вони є глобальними змінними, тобто тими реальними значеннями, з якими будуть зроблені обчислення.

1.5.2. Редагування процедур

Створення процедури

Існують два способи створення процедури:

- активізувати пункт головного меню **Edit** (Редагування). Вибрати **New Sub** або **New Function** (Нова Підпрограма або Нова Функція); при цьому відкриється діалогове вікно. Ввести ім'я нової процедури, наприклад, **Sub Sum(x,y**) і натиснути клавішу **Enter.** Після цього можна вводити оператори тіла процедури;
- набрати текст основної програми. Ввести ключове слово **Sub** або **Function,** після якого вказати ім'я процедури і список формальних параметрів (якщо він є), наприклад, **Sub Sum(x,y**). Відкриється нове вікно, у якому будуть два рядки: набраний Вами (**Sub Sum(x,y**)) і **End Sub** або **End Function**.

Після цього можна вводити оператори тіла процедури.

Ім'я процедури повинно бути унікальним і не повторювати імен інших процедур і змінних.

Збереження процедури

При збереженні програми (**Save** або **Save As**) QBasic записує тексти основної програми і всіх процедур в один файл. При цьому він автоматично додає в початок основної програми оператори **DECLARE**, які містять назви використовуваних процедур і повідомляють про їх наявність. Таким чином, основна програма завжди буде починатися з декларації всіх процедур (підпрограм і функцій), які викликаються нею.

Перегляд списку процедур

Для перегляду списку процедур, які використовуються головною програмою, треба активізувати пункт меню **View** (Перегляд), вибрати пункт **SUBs** і натиснути клавішу **Enter**. Це можна зробити також, натиснувши клавішу **F2**. При цьому з'явиться діалогове вікно зі списком всіх програмних елементів, включаючи головну програму.

Виконати багатомодульну програму можна, натиснувши клавішу **F5**, перебуваючи в основній програмі або в будь-якій процедурі.

1.5.3. Процедури - функції або функції користувача (FUNCTION)

Поряд зі стандартними вбудованими функціями (SIN, EXP, LOG та ін.) QBasic надає можливість користувачеві створювати свої функції.

Як і стандартні вбудовані функції, функції користувача, якщо потрібно, отримують аргументи через список параметрів або загальну область і повертають в основну програму обчислене значення.

Функція повертає єдине значення в основну програму, тому всі функції повинні містити хоча б один оператор присвоювання певного значення імені функції. Присвоєння може відбуватися кілька разів при наявності, наприклад, умовного оператора, але при кожному виклику функція повертає тільки один результат.

Функція обчислюється в момент звертання до неї за іменем

ім'я [(фактичні параметри)]

Загальна форма запису

FUNCTION *ім'я* **[(***формальні параметри***)]** [**STATIC**]
[*блок_операторів*]
ім'я *= вираз*
[EXIT FUNCTION]
[*блок_операторів*]
END FUNCTION

} тіло функції

де *ім'я* - ім'я функції користувача. Тип даних, які повертаються нею, може бути визначений явно суфіксом типу даних (%, &, !, # або $).

- *формальні параметри* - одна або кілька змінних, які вказують параметри, передані у функцію при її виклику;

мають вигляд:

- *змінна*[( )] [AS *тип*] [, *змінна*[( )] [AS *тип*]]...;
- *змінна* - ім'я змінної Qbasic;
- *тип* - тип *змінної* (INTEGER, LONG, SINGLE, DOUBLE, STRING або тип даних, які визначені користувачем);
- **STATIC** – вказує на те, що значення локальних змінних зберігаються між викликами функції;
- *вираз* - значення функції, що повертається. Воно присвоюється імені функції як локальній змінній і повинне бути з ним одного типу (за замовчуванням це речовинний тип);

Кожна процедура повинна мати початок і кінець. Оператором **END FUNCTION** завершується робота процедури, і управління передається оператору основної програми, розташованому за оператором виклику даної процедури.

Оператор **EXIT FUNCTION** дозволяє перервати виконання процедури і передати управління в основну програму.

Допоміжні змінні в процедурі мають локальний характер. Для них автоматично резервується пам'ять, і їхні значення в основну програму не повертаються.

Команда **SHARED <***список змінних***>**
визначає зазначеним у списку змінним ті ж комірки пам'яті, що були призначені головною програмою; у такий спосіб значення цих змінних стають доступними головній програмі (ці змінні не повинні бути параметрами).

Якщо в програмі зустрічається звертання до функції, то виконуються такі дії:

1) обчислення значень фактичних параметрів;
2) звертання до команд опису даної функції;
3) присвоювання формальним параметрам значень фактичних і виконання тіла функції;
4) присвоювання необхідного значення імені функції
5) повернення обчисленого значення в програму, яка викликала функцію.

Приклад. Розглянемо використання функції, визначеної користувачем, на прикладі обчислення числа можливих сполучень

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

Як видно з формули, треба тричі обчислювати значення факторіала, для чого визначимо функцію користувача Fact () і тричі використаємо її в основній програмі.

```
DECLARE FUNCTION FACT% (K AS INTEGER) ’Оголошення функції
REM Основна програма
DIM M AS INTEGER, N AS INTEGER: ’Оголошення змінних
INPUT ’’Задайте n, m’’,n,m :‘Задання початкових даних
'Фактичними параметр. при виклику функції Fact() є n, m і n-m
C = Fact%(n) / (Fact%(m) * Fact%( n-m)):’Обчислення формули
PRINT ’’Кількість сполучень = ’’,C:’Виведення результату
END
'Кінець основної програми

FUNCTION FACT% (K AS INTEGER) 'K - формальний   параметр функції
REM Функція обчислення факторіала
DIM I AS INTEGER, F AS INTEGER
IF K<0 THEN PRINT “факторіал ”; K; “ не існує”:STOP
IF K = 0 THEN
    Fact%=1: ’Присвоювання значення, що повертається
ELSE
    F = 1'Обчислення факторіала
    FOR I = 1 TO K
        F =F * I
    NEXT I
    FACT% = F ’Присвоювання значення, що повертається
END IF
END FUNCTION
```

1.5.4. Підпрограми (SUB)

Підпрограма - це процедура, що викликається з основної програми за допомогою оператора виклику **CALL**. У головну програму при цьому можна повертати не одне (як при стандартному використанні **FUNCTION**), а кілька обчислених значень.

Загальна форма опису підпрограми SUB:

SUB *ім'я*[(*формальні параметри*)] **[STATIC**]
[*блок_операторів*]
[EXIT SUB] } тіло підпрограми
[*блок_операторів*]
END SUB

ім'я - ім'я підпрограми **SUB**, довжиною до 40 символів, без суфікса типу даних. Інші складові загальної форми опису аналогічні опису **FUNCTION.**

Приклад.

З використанням процедури-підпрограми підраховується вартість телефонної розмови із щохвилинною оплатою: 0.22 грн.+20% податку

```
DECLARE SUB CINA (K,C)
CLS
INPUT “Кількість хвилин розмови з містом Київ===> “,K1
CALL CINA (K1,C)
```

```
PRINT “Вам необхідно сплатити за розмову з Києвом ==>”;C; “грн.”
INPUT “Кількість хвилин розмови зі Львовом ===> “,K2
CALL CINA (K2,C)
PRINT “ Вам необхідно сплатити за розмову зі Львовом ===>”;C; “грн.”
END

SUB CINA (K,C)
C=0.22*K
C=C+0.2*C
END SUB
```

Використання ключового слова **STATIC** робить всі локальні змінні статичними, тобто вони не будуть ініціалізуватися знову при наступному виклику даної підпрограми.

Приклад.

```
DECLARE SUB CALC
FOR TIME%=1 TO 4
    CALL CALC
NEXT TIME%
END
SUB CALC STATIC
NUMBER%=NUMBER%+1
PRINT NUMBER%
END SUB
```

Результат виконання програми зі STATIC	Результат виконання програми без STATIC
1	1
2	1
3	1
4	1

1.5.5. Передача параметрів у процедуру

Існують два основні правила передачі параметрів у процедури:

- кількість фактичних параметрів при звертанні до процедури повинна дорівнювати кількості формальних параметрів у її оголошенні;
- відповідні параметри повинні бути одного типу.

У дужках при виклику процедури можна вказувати як константи (числові, символьні), так і значення змінних або самі змінні. Тому існують два основних засоби передачі параметрів - за значенням і за посиланням.

Передача параметра за посиланням

Як фактичний параметр передається не значення змінної, а її адреса. Якщо значення формального параметра змінюється в процедурі, то фактичний параметр набуває цього значення.

Таким чином, процедура отримує доступ до елемента пам'яті, у якому зберігається значення змінної, і може змінити її значення

Передача параметра за значенням

У процедуру передається копія значення змінної, яка зберігається в тимчасовій області пам'яті. Відповідний параметр

при виклику процедури береться в круглі дужки. Це доцільно робити тоді, якщо процедура не повинна змінювати значення змінних, що передаються їй.

Якщо фактичний параметр є константою, QBasic не може передати його по посиланню, тому що він не має адреси.

Якщо за значенням передається змінна, то вона є локальною змінною процедури, якій присвоєне значення фактичного параметра. Тому зміна цієї локальної змінної не впливає на значення змінної в основній програмі.

Передача масиву

Масив або елемент масиву завжди передаються в процедуру за посиланням. Ім'я масиву або елемента завжди вказує адресу фактичного параметра, що дозволяє процедурі змінювати їхні значення.

Наявність порожніх дужок після імені вказує, що передається масив, а не скалярна змінна. При цьому формальний параметр теж повинен містити круглі дужки.

Якщо потрібно передати частину масиву, то потрібно додати параметри, що вказують границі цієї частини.

Якщо потрібно передати елемент масиву - то треба вказати номер елемента в дужках після імені масиву.

Стандартні функції **LBOUND(A)** і **UBOUND(A)** визначають найбільше й найменше значення індексів масиву А.

Приклад.

```
SUB P2(A(),S)
S=0:FOR I= LBOUND(A) TO UBOUND(A)
    S=S+A(I)
NEXT I
END SUB
```

Приклад.

Задані масиви $\{a_i\},\{b_i\},i=1,k$ Знайти індекс мінімального елемента масиву $\{c_i\},i=1,k$, де $c_i = a_i - b_i$

У головній програмі здійснюється введення вихідних масивів, виклик підпрограми і виведення повернутого з підпрограми значення індексу мінімального елемента масиву $\{c_i\}$

```
DECLARE SUB MININD (A!(), B!(), C!(), N AS INTEGER, IMIN%)
DIM K AS INTEGER, I AS INTEGER
CLS
INPUT “Кількість елементів К= “,K
DIM A(K), B(K),C(K)
FOR I=1 TO K
PRINT “A(“;I;”)=”; : INPUT A(I)
```

```
NEXT I
FOR I=1 TO K
PRINT “B(“;I;”)=”; : INPUT  B(I)
NEXT I
CALL MININD(A(),B(),C(), K, KMIN%)
PRINT ” Індекс мінімального елемента =”;KMIN%
END

SUB MININD (A!(), B!(), C!(), N AS INTEGER, IMIN%)
DIM I AS INTEGER
FOR I=1 TO N
    C(I)=A(I)-B(I)
NEXT I
CMIN=C(1)
IMIN%=1
FOR I=2 TO N
    IF C(I)< CMIN THEN CMIN=C(I):IMIN%=I
NEXT I
END SUB
```

1.5.6. Використання функцій типу DEF FN (нестандартні функції користувача)

DEF FN є альтернативною формою функції, розробленою користувачем.

Ці функції можуть викликатися тільки в програмі, де вони описані. QBasic підтримує два різновиди функції типу **DEF FN:** однорядкову й багаторядкову.

Подібно **FUNCTION**, **DEF FN** повертає значення в основну програму. Однак при цьому є невід'ємною частиною основної програми.

Формат функції:

Однорядкова (проста форма**):**

DEF FNім’я (параметри) = вираз

Багаторядкова (складна форма**):**

DEF FNі*м’я* (*параметри*)
оператори
FNі***м’я*** =*вираз*
оператори
END DEF

де *ім'я* – ім'я функції;
параметри – список формальних параметрів;
вираз – вираз, значення якого присвоюється функції;
оператори – тіло функції.

FNім’я використовується для визначення функції й повинне характеризувати який-небудь тип даних. Параметри не можуть містити масиви або символьні константи. Тип виразу повинен відповідати типу функції.

У багаторядковій формі **DEF FN** для переривання виконання функції використовується оператор **EXIT DEF**, аналогічний за своєю дією операторові **EXIT FUNCTION**.

Звертання до нестандартної функції

FN*ім’я* (*параметри*)

де *ім'я* – ім'я функції;

параметри – список фактичних параметрів

При звертанні до функції обчислюється значення виразу в операторі **DEF**, при цьому формальні параметри заміняються відповідними фактичними значеннями, які повинні відповідати формальним за кількістю, порядком слідування і типом.

Приклад.

Обчислити $\omega = \sqrt{x^2 + y^2 + \sin^2 xy} + \sqrt{y^2 + z^2 + \sin^2 yz} + \sqrt{z^2 + x^2 + \sin^2 xz}$

при x=21,87 y=35,13 z=9.87

```
DEF FNF (A, B) =SQR(A*A+B*B+Sin (A*B)(2)
DATA 21.87, 35.13, 9.87
READ x, y, z
W=FNF (x,y)+FNF (y,z)+FNF (z,x)
```

Приклад.

(однорядкова функція викликається в операторі PRINT).

```
DEF FNVS (A!, B!) = (A! + B!)/2
PRINT FNVS (8.5, 44.)
```

1.6. ФАЙЛИ НА МАГНІТНИХ НОСІЯХ

Багато програм використовують дані, що вводяться за допомогою клавіатури клавіатури: обробляють їх, зберігають у масивах і змінних, виводять безпосередньо на екран. Однак у всіх програмах дані зникають із пам'яті комп'ютера, як тільки програма закінчує свою роботу.

Основним способом збереження інформації, отриманої в ході виконання програми, служить запис її на тверді або гнучкі диски. Це особливо важливо, якщо обсяг інформації великий, а дані передбачається використати неодноразово в цій або в інших програмах.

Використання файлів дає такі переваги:

1) можливість зберігати дані на зовнішніх носіях тривалий час, а не тільки під час виконання програми;

2) кількість даних може бути великою, заздалегідь невідомою, тобто під час роботи з файлом не потрібно знати число записів у ньому;

3) дані в записі можуть бути різних типів;

4) дані зберігаються окремо від програм, які їх обробляють, і можуть бути використані різними програмами й користувачами;

5) між даними різних файлів може бути певна відповідність (релятивістські зв'язки), що дає можливість не дублювати ті самі дані в різних файлах.

Інформація, розміщена на зовнішньому носії, що має певну логічну структуру та ім'я, називається **набором даних**. Він може містити як програми на вхідних мовах, так і оброблені дані (каталог книг бібліотеки, відомості про абітурієнтів академії і т.п.).

Набір даних складається з фізичних записів.

Фізичний запис – найменша порція даних, що переноситься із магнітного диска (МД) або на МД за одну фізичну операцію введення-виведення. Вона являє собою одиницю носія - сектор МД (512 Б).

Файл – символічне подання набору даних як сукупності логічних записів однакової структури.

Логічний запис – змістовна одиниця збереженої у файлі інформації, що є сукупністю відомостей про деякий об'єкт.

Приклад. Файл - список студентів групи, логічний запис - містить поля: прізвище, ім'я студента, рік народження.

Розмір логічного запису залежить від розв'язуваної задачі. Файл зв'язується з набором даних при так званому **відкритті** файлів у програмі.

Кількість записів у файлі може бути довільною. За останнім записом знаходиться службова мітка кінця файла (код 26). Її записує у файл QBasic. Файл може бути порожнім, тобто в ньому не буде жодного запису, але ім'я файла і мітка кінця будуть обов'язково. Записи можуть містити дані різних типів.

Приклад. **" Іванов", "Петро", 1986**

Такий запис містить 3 поля-параметри:

- прізвище - текстовий тип;
- ім'я - текстовий тип;
- рік народження - цілий числовий тип;

Файли розрізняються відповідно до **методу доступу** до інформації в них, незалежно від типу збережених у них даних.

Метод доступу визначає порядок, у якому дані записуються у файл або читаються з файла.

Підтримуються такі методи доступу до інформації у файлах:

– **послідовний доступ** (текстові файли);

– **прямий (довільний**) доступ.

Файли **послідовного доступу** дозволяють прочитати деякий запис, тільки прочитавши всі попередні. Їх можна порівняти з аудіо- або відеозаписами на магнітофонній стрічці: для пошуку потрібного місця ви змушені перемотувати стрічку і послідовно її переглядати або прослуховувати.

Інформація в них зберігається в текстовому форматі (у вигляді ланцюжка кодів ASCII).

Записи в такому файлі легко проглядаються просто з екрана. Такі файли можуть оброблятися будь-якими текстовими редакторами так само, як і самим QBasic. Але, оскільки інформація зчитується й зберігається послідовно, це робить процес обробки інформації в них більш тривалим у порівнянні з обробкою файлів інших типів.

Файли **прямого доступу** зберігають інформацію в спеціальному форматі. Кожен запис займає певну (однакову для всіх записів) довжину, тому файли з таким доступом можуть займати більше місця на диску, зате робота з ними відбувається швидше. Звертання до записів у таких файлах здійснюється "*прямо*" - без читання попередніх даних, за номером запису (шифр книги в бібліотеці). Записи можуть бути будь-якої довжини(до 32767 байтів).

Будь-який файл можна організувати так, щоб мати або прямий, або послідовний доступ, але не обидва разом.

Якщо при кожному звертанні до файла використовуються майже всі дані, а змінювати їхній вміст часто не передбачається, то доцільно вибрати послідовний доступ. Коли потрібно часто змінювати вміст записів і переглядати їх у довільному порядку - прямий доступ.

Крім цього, QBasic дозволяє звертатися до файла як до послідовності байтів і оперувати безпосередньо байтами. Такий спосіб доступу до інформації називається **бінарним (**теж прямого доступу, але обіг за номером байта).

Будь-які дії з файлами, які виконуються в програмі, містять у собі такі обов'язкові кроки:

- відкриття файла;
- читання й запис оброблюваних даних;
- закриття файла.

1.6.1. Файли послідовного доступу

Щоб **створити** послідовний файл, необхідно:

1) відкрити файл оператором **OPEN** для виведення або дозапису в нього даних;

2) писати дані у файл операторами **PRINT** або **WRITE**;

3) закрити файл оператором **CLOSE**.

Щоб **прочитати** послідовний файл, необхідно:

1) для доступу до даних відкрити його оператором **OPEN** для виведення з нього даних;

2) читати дані з файла в програму операторами **INPUT**, **LINE INPUT** або **INPUT$**,

3) закрити файл оператором **CLOSE**.

Можна відкрити кілька файлів одночасно, щоб читати, наприклад, інформацію з одного файла, обробляти її і потім зберігати в іншому. Але не можна використовувати відкритий файл і для читання, і для запису інформації одночасно, тому що ці дії пов'язані з різними режимами відкриття файла.

Для **відкриття** файла використовується оператор **OPEN**, що має такий формат:

OPEN *файл$* **FOR** *режим* **AS** *#номер_файла %*
де *файл$* - ім'я файла;
режим - режим роботи з даним файлом;
- необов'язковий знак, що випереджає номер файла;
номер_файла % **(**дескриптор**)** - ціле число від 1 до 255.

Приклад.
OPEN "*Рік народження.txt* " FOR OUTPUT AS #5

Параметри оператора **OPEN:**

- Параметр *файл$*. Кожний файл повинен мати ім'я, складене за певними правилами, щоб операційна система комп'ютера знала, де шукати той або інший файл. Тому повне ім'я файла включає не тільки безпосередньо назву, але й найменування дискового пристрою і директорії, де знаходиться або буде створений файл.

Якщо файл, необхідний вам у програмі, знаходиться на тому же диску, що й QBasic, але в іншій директорії, необхідно все рівно вказувати повний шлях до нього.

Приклад.
Якщо файл Рік народження**.**txt знаходиться в піддиректорії MY_DATA\ директорії DOS\ диску C**:,** він може бути відкритий у програмі за допомогою такого оператора:
OPEN "C:\DOS\MY_DATA\ *Рік народження.txt* " FOR INPUT AS #2

Найпростіший спосіб роботи з файлами - зробити поточною ту піддиректорію, у якій знаходяться файли, що цікавлять вас, і потім просто вказувати їхні імена.

Як параметр *файл$* можна використати змінну текстового типу, якій присвоєне ім'я файла. Це дозволяє вводити ім'я файла з клавіатури і, таким чином, використовувати програму для роботи з різними файлами. У цьому випадку оператор відкриття файла набуває виду:

Приклад.

```
INPUT "Введіть ім'я файла: ", FileNames$
                OPEN FileNames$ FOR APPEND AS #1
```

Такий метод відкриття файлів є найбільш універсальним, оскільки він простий і зручний для користувача.

- Параметр *режим* визначає режим роботи з файлом, тобто вказує QBasic, як ви збираєтеся цей файл використати: для читання або запису в нього інформації. Цей параметр може мати одне з таких значень:

OUTPUT – відкрити новий файл для запису в нього інформації. Якщо відкрити в цьому режимі вже існуючий файл, його вміст буде стерто;

APPEND – відкриває існуючий файл для додавання в нього нової інформації. У такому режимі нова інформація завжди міститься в кінці файла, після останнього запису. Якщо вказати ім'я файла, якого ще не існує, буде відкритий новий файл, як це робиться в режимі **OUTPUT**;

INPUT – відкриває існуючий файл для читання збереженої в ньому інформації. У цьому випадку, якщо вказати ім'я неіснуючого файла, отримаємо повідомлення про помилку **File not found** (Файл не знайдений);

- Параметр *номер_файла%* присвоює файла, що відкривається, певний номер для більш зручного використання в програмі.

У прикладі з файлом з ім'ям ***Рік народження.txt*** йому присвоюється номер 2. Після цього даний номер не можна буде присвоїти ніякому іншому файла, поки ***Рік народження.txt*** відкритий.

Щоб працювати з файлами, потрібно розуміти, як зв'язується операційна система з файлом. Для цього є канал введення-виведення. При відкритті файла ставиться у відповідність канал з певним номером. Отже, при роботі з файлом має значення не ім'я файла, а номер каналу. Операційна система повинна мати відомості про наявність вільних каналів.

Число одночасно відкритих файлів у програмі залежить від конфігурації операційної системи, у якій задається максимальне число файлів, що одночасно відкриваються. Можна змінити це число за допомогою команди **FILES** у файлі **CONFIG.SYS**, що визначає конфігурацію вашої системи.

Для запобігання хаосу, не слід відкривати занадто багато файлів одночасно. Як тільки закінчена робота з яким-небудь файлом, відразу треба його закрити.

Для **занесення даних** у файл (формування запису на зовнішньому носії) використовують команди **PRINT** і **WRITE**.

Формат операторів:

PRINT #*номер_файла*%**, [***список значень***]**

WRITE # *номер_файла***%, [***список значень***]**

список значень - список констант або/і змінних, значення яких записуються у файл. Якщо список значень відсутній, у файл буде занесений порожній рядок. Логіка роботи цих операторів різна.

Оператор **WRITE**

Роздільником у списку значень є **кома.** Елементи списку заносяться у файл в один текстовий рядок. Символьні дані записують у лапках. Ця команда сама проставляє коми між полями і більш ощадливо розташовує дані на носії. Після запису останнього елемента рядка автоматично записуються символи "повернення каретки" (код 13) і "перехід на новий рядок" (код 10). Дані можна прочитати оператором **INPUT.**

Оператор **PRINT**

Роздільниками в списку значень є:

кома - значення записуються в 14-символьні зони виведення (після кожного даного автоматично заноситься символ табуляції);

крапка з комою - значення записуються підряд, без проміжків між ними.

Оператор **PRINT** зручний для ретельного редагування тексту вихідного файла, а оператор **WRITE** краще застосовувати в тому випадку, коли вихідний файл буде використовуватися надалі як вхідний для інших програм.

Приклад. Програма запису у файл із ім'ям ***Рікнародження.txt*** таких даних (рядків)

Іванов	**Петро**	**1986**
Сурина	**Марина**	**1987**

```
Dim Fam As String, Im As String, God As Integer
Open " Рікнародження.txt" for Output As#1
For i=1 to 2
    Input "Прізвище"; Fam
    Input "Ім'я"; Im
    Input "Рік народження"; God
    Write #1, Fam, Im, God
Next i
Close #1
"Іванов", "Петро", 1986
"Сурина", "Марина", 1987
```

Замінимо **Write #1, Fam, Im, God**
оператором **Print #1, Fam, Im, God**

```
1          15          30
Іванов     Петро       1986
Сурина     Марина      1987
```

А тепер оператором **Print #1, Fam; Im; God**

```
1      7       13
ІвановПетро  1986
СуринаМарина 1987
```

Можна вставити коми в лапках у список значень **PRINT**, щоб розбити запис на більш дрібні частини й забезпечити доступ до цих частин (у майбутньому) командою читання.

Дані можна записувати не тільки в змінні, але й у заданий блок оперативної пам'яті.

Для **закриття** файла використовується оператор:

CLOSE [*#номер_файла*]

Якщо вказується номер файла, то буде закритий саме цей файл.

Оператор **CLOSE** без параметра закриває всі файли, відкриті в цей момент у програмі.

Читати дані можна різними способами:

- за допомогою операторів **INPUT** і **LINE INPUT;**
- за допомогою функції **INPUT$.**

Формат цих операторів

INPUT*# номер_файла***%,** *список змінних*

список змінних – записані через кому змінні будь-якого типу. Змінні списку набувають відповідних значень полів запису файла.

LINE INPUT*# номер_файла***%,** *змінна*

змінна – змінна типу String

Зчитує рядок до 255 символів з файла (поки не зустріне символ "ВК") і присвоює *змінній* це значення

INPUT$(*n***[,#***номер_файла***%])**

n – число символів (байтів) для читання;

змінна – змінна типу String

Зчитується певна кількість символів із зазначеного файла

Приклад.

```
INPUT                          LINE INPUT                     INPUT$
Do Until EOF(1)                Do Until EOF(1)                Input$ (LOF(1),#1)
Input #1, Fam, Im, God             Line Input #1, text$       Close #1
Loop                           Loop
Close #1                       Close #1
```

EOF *(номер_файла)* - повертає ІСТИНА при досягненні кінця файла.

LOF *(номер_файла)* – повертає довжину файла

1.6.2. Файли довільного доступу

Щоб **створити довільний файл** необхідно:

1) оператором **OPEN** відкрити файл, указуючи довжину запису.

Формат

OPEN *файл$* **FOR Random [Access Доступ][блокування] AS** #*номер_файла* **% Len=***довжина_запису*

На відміну від текстових файлів, тут не існує різниці між файлами для запису й для читання: всі вони відкриваються в одному режимі, що визначається ключовим словом **Random.**

Крім цього, потрібно вказати довжину запису атрибутом **Len.** Якщо це значення менше реальної довжини запису, то виникає помилка, якщо більше, то при записі файла використовується більше дискового простору, ніж необхідно.

Існує можливість відкрити файл із видом доступу “тільки читання” або забороною читання і запис в нього інформації за допомогою програм інших комп'ютерів, або встановити захист на окремий запис за номером;

2) оператором **FIELD**, розподілити простір у буфері прямого доступу для змінних, значення яких будуть записані у файл. При цьому запис буде мати необхідну структуру.

Формат

FIELD #*номер_файла, довжина поля1* **As** *змінна1***[,***довжина поля2* **As** *змінна2***...]**

Сумарна довжина всіх перерахованих полів повинна збігатися з обсягом буфера, оголошеним при відкритті файла;

3) всі числові дані, які записуються у файл довільного доступу, попередньо повинні бути перетворені до символьного типу. Це перетворення виконують функції **MKI$, MKL$, MKS$, MKD$** (Make...). Значення аргументів не повинні виходити за діапазон відповідних числових типів. Довжина рядків, що повертаються 2, 4, 4, 8 відповідно. Тільки після цього дані можна помістити в буфер.

LSET і **RSET** переміщають дані в буфер прямого доступу, підготовляючи його до запису у файл і здійснюють відповідно ліве або праве вирівнювання символьних значень.

4) після заповнення буфера його вміст можна перенести у файл оператором PUT.

Формат:

PUT *#номер_файла%* **[,** *номер запису*]**[,***змінна***]**
#номер_файла% - номер відкритого файла;
номер запису - номер запису, в якому будуть розміщені дані;
змінна - містить дані для запису у файл;

5) закрити файл оператором **CLOSE.**

Приклад.

Для наведеного вище прикладу, у припущенні, що прізвище містить до 15 символів, ім’я - до 10 символів, рік народження - 4 цифри

```
Dim Fam As String, Im As String, God As Integer
Dim Fam1As String, Im1 As String, God1 As String
Open “Рік народження.txt” for FOR Random AS #1 % Len=27
FIELD #1 15 As Fam1, 10 As Im1, 2 As God1
For i=1 to 2
    Input “Прізвище”; Fam
    Input “Ім'я”; Im
    Input “Рік народження”; God
    Write #1, Fam, Im, God
        LSET Fam1, Fam
        LSET Im1, Im
        LSET God1, MKI$(God)
    PUT #1, Fam1,Im1,God1
Next i
CLOSE #1
```

Щоб **прочитати довільний** файл, необхідно:

1) відкрити файл оператором **OPEN**, вказуючи довжину запису;

2) оператором **FIELD**, розподілити простір у буфері прямого доступу для змінних, значення яких будуть читатися з файла;

3) помістити необхідний запис у буфер оператором **GET**.

Формат оператора **GET:**

GET *#номер_файла%* **[,** *номер запису*]**[,***змінна*]
#номер_файла% - номер відкритого файла;
номер запису - номер запису для зчитування даних;
змінна - містить дані, що зчитуються з файла.

Дані в буфері доступні програмі. Символьні дані при необхідності повинні бути перетворені в числові функціями **CVI, CVL, CVS, CVD** (Convert...);

4) закрити файл оператором **CLOSE**

Відзначимо такі корисні оператори для роботи з наборами даних:

Оператор **перейменування** файла
NAME *старе ім'я* **AS** *нове ім'я*

Оператор **видалення** файла
KILL *ім'я файла, що видаляється*

1.7. ОБРОБКА СИМВОЛЬНИХ ДАНИХ

Для розуміння символьних і текстових функцій необхідно попередньо згадати кодову систему, яка використовується комп'ютерами.

1.7.1. Одержання ASCII - коду символу й одержання символу, що відповідає ASCII - коду

Для виконання цих перетворень в QBasic передбачені такі функції:

ASC (*текст***)** – повертає код ASCII для першого символу текстового значення;

CHR$ (*код***)** – повертає символ, що відповідає коду ASCII.

Параметр *код* повинен мати значення від 0 до 255 і бути одним з ASCII - кодів.

Приклад

```
PRINT "Символ з кодом ASCII 81 - це "; CHR$(81)
PRINT "Код ASCII символу "Q" - це "; ASC ("Q")
```

Результат

```
Символ з кодом ASCII 81- це Q
Код ASCII символу "Q" - це 81
```

1.7.2. Введення символів у програму

Оператор **INPUT** дозволяє ввести символьні вирази будь-якої припустимої довжини (у лапках або без них), при цьому

завершення введення відбуваються при натисканні клавіші ENTER.

При використанні функції **INPUT$** ви не повинні натискати клавішу ENTER після введення символу – введення завершується відразу після одержання символу.

Функцію **INPUT$** можна застосовувати й для введення в програму декількох символів.

Загальний вид:

INPUT$ (*число***),** де *число* – кількість символів для введення.

Символи, що вводяться, не відображаються на екрані.

Приклад

```
PRINT "Введіть дві літери"
COD$=INPUT$(2)
PRINT "Введені літери";COD$
```

Оператор **LINE INPUT** дозволяє ввести довільний текстовий рядок до 256 символів, включаючи коми. Можна вводити будь-які символи, не перевіряючи, чи є вони літерами.

Приклад

```
PRINT "Введіть будь-який текст"
LINE INPUT COD$
PRINT "Введений  текст";COD$
Введіть будь-який текст
Нг89б;;.?fs
Введено  текст Нг89б;;.?fs
```

1.7.3. Підтримка інтерфейсу між програмою та клавіатурою (INKEY$)

Припустимо, необхідно створити паузу очікування в певному місці вашої програми, наприклад, щоб мати час для прочитання великого повідомлення (це може бути вікно з рекомендаціями з користування програмою). Звичайно в подібних ситуаціях програма чекає натискання певної або довільної клавіші.

Функція **INKEY$** аналізує системну інформацію про натиснуті клавіші і може бути використана для створення паузи довільної тривалості. Вона повертає порожній рядок, якщо не була натиснута жодна клавіша, або рядок з одного символу при натисканні звичайної клавіші, або рядок із двох символів при натисканні клавіші з розширеним кодом (128-255). При цьому символ, що вводиться із клавіатури, на екран не виводиться.

Щоб організувати таку паузу, потрібно використати умовний цикл, у якому перевіряється значення, що повертається функцією **INKEY$.**

У наведеному нижче прикладі програма буде знаходитись в режимі очікування до моменту натискання довільної клавіші.

Приклад

```
PRINT "Press any key to continue"
DO WHILE INKEY$ = ""
LOOP
```

Однак, можна задати й конкретну клавішу, при натисканні на яку виконання програми буде продовжено:

Приклад

```
PRINT "Press ' ESC ' to continue"
DO WHILE INKEY$ <> CHR$(027)
LOOP
```

1.7.4. Визначення довжини текстового виразу

Функція **LEN** призначена для визначення довжини символьного виразу

LEN (*текст***),**

де *текст* – текстовий вираз.

Функція обчислює довжину тексту, тобто число символів (включаючи проміжки).

Приклад

```
PRINT LEN("ВАСЯ")
```

***результат*: 4**

1.7.5. Вибір підрядка (виділення частини тексту)

У мові QBasic існує кілька функцій, що дозволяють виділяти символи з текстового виразу. Ці функції мають такі формати:

LEFT$ (*текст, число***)**
RIGHT$ (*текст, число***)**
MID$ (*текст, позиція*[**,***число*]**),**

де *текст* – текстовий вираз;
число – число виділюваних символів (від 0 до 32767);
позиція – номер символу, з якого починається виділення.

Функція **LEFT$** повертає підрядок довжиною *число*, що починається з першого символу вхідного рядка (з лівого краю тексту).

Функція **RIGHT$** повертає зазначене *число* символів, починаючи із правого краю тексту, аналогічно функції LEFT$.

Функція **MID$** повертає підрядок довжиною *число*, починаючи з *позиції* від лівого краю тексту. Для функції **MID$** параметр *число* не є обов'язковим. Якщо він не зазначений, то функція повертає правий залишок рядка від заданої *позиції* до кінця текстового виразу.

Якщо параметр *число* дорівнює 0, то всі функції повертають порожній рядок. Якщо значення даного параметра більше, ніж реальна довжина тексту, буде повернуто весь текстовий вираз.

Приклад	***Результат***
CLS **EX$ = "Бажаю все знати"** **PRINT LEFT$ (EX$, 5)** **PRINT RIGHT$ (EX$, 9)** **PRINT MID$ (EX$, 7 , 3)**	Бажаю все знати все

1.7.6. Пошук підрядка в символьному виразі

Припустимо, що маємо великий текстовий файл і бажаємо знайти місце розташування певної фрази в ньому. Програма істотно спроститься при використанні функції **INSTR**, що спеціально призначена для пошуку певного підрядка в символьному виразі.

Функція **INSTR** повертає числове значення, що вказує позицію, з якої починається підрядок, що шукається, відносно початку тексту.

INSTR ([*позиція***,]** *текст***,** *підрядок***),**

де *позиція* – номер символу, з якого починається пошук підрядка;
текст – текстовий вираз, у якому відбувається пошук;
підрядок –текстовий вираз, що шукається.

Параметр *позиція* необов'язковий, якщо його опустити, пошук буде виконуватися з першого символу текстового виразу.

Функція **INSTR** повертає 0, якщо:

- підрядок не знайдено;
- значення параметра *позиція* більше довжини тексту;
- довжина тексту нульова.

Якщо підрядок порожній, функцією буде повернуте значення *позиція* (якщо параметр не визначений, то буде повернута 1). Пошук припиняється при першому ж знаходженні підрядка, тому друге входження підрядка знайдене не буде.

Приклад

```
CLS
EX$ = "АНЯ,ЯН,АН-ДРУЗІ"
PRINT EX$
PRINT "ІМ'Я ' ЯН ' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ"; INSTR (EX$, "ЯН")
PRINT "ІМ'Я 'АНЯ' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ";INSTR (EX$, "АНЯ")
PRINT "ІМ'Я 'АН' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ";INSTR (EX$,"АН")
```

Результат

```
АНЯ,ЯН,АН-ДРУЗІ
ІМ'Я ' ЯН ' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ 5
ІМ'Я 'АНЯ' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ 1
ІМ'Я 'АН' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ 8
```

1.7.7. Приведення тексту до різних варіантів написання

Якщо текст написаний маленькими літерами, а ви хочете вивести його на екран і роздрукувати, використовуючи тільки великі, вам не треба вводити текст знову - скористайтеся першою з таких функцій:

UCASE$ (*текст***)**

LCASE$ (*текст***),**

де *текст* – текстовий вираз, призначений для зміни регістру.

Функція **LCASE$** перетворить усі літери рядка в маленькі, функція **UCASE$** -у великі.

Приклад

```
BAL$="рахунок:"
GAL$="Нуль"
PRINT BAL$;GAL$
PRINT
PRINT UCASE$ (BAL$);
PRINT LCASE$ (GAL$)
```

Результат

```
рахунок: Нуль
РАХУНОК: нуль
```

Ці функції корисні при перевірці відповідей на питання програми типу "Бажаєте повторити? Y/N ".

У тому випадку, коли невідомо, малу або велику літеру введе користувач, кожна із цих функцій замінить подвійну перевірку.

Приклад

```
BEGIN:
……..
INPUT "Бажаєте повторити? Y/N", IFLAG$
IF UCASE$(IFLAG$)= "Y" THEN BEGIN
```

1.7.8. Формування тексту і видалення початкових або кінцевих проміжків

Функція **SPACE$(***N***)** повертає рядок, що містить ***N*** проміжків.

Приклад	*Результат*
	1
FOR I=1 TO 5	2
PRINT SPASE$(I);I	3
NEXT I	4
	5

Функція **STRING$(*N*,** *символ***)** повертає рядок, що містить ***N*** одиночних символів *символ* (замість символу можна вказати його код від 0 до 255).

Приклад	*Результат*
PRINT STRING$(5,42);"QBASIC";STRING$(5,"/")	*****QBASIC/////

Функція **LTRIM$ (***текст***)** повертає рядок, отриманий з рядка *текст* видаленням з нього початкових (лівих) проміжків.

Функція **RTRIM$ (***текст***)** повертає рядок, отриманий з рядка *текст* видаленням з нього кінцевих (правих) проміжків.

Приклад	*Результат*
A$=" QBasic "	
B$=LTRIM$(A$)	
PRINT A$;"*"	QBasic *
PRINT B$;"*"	QBasic *
B$=RTRIM$(A$)	
PRINT B$;"*"	QBasic*

1.7.9. Перетворення текстових значень у числові й навпаки

Функція **VAL(***текст***)** повертає числове значення аргументу *текст*, що має символьне значення арифметичного виразу.

Приклад	*Результат*
PRINT VAL ("3.14")	3.14

Функція **STR$(***число***)** повертає символьне подання числа або числового виразу.

Приклад	*Результат*
PRINT STR$ (123)	"123"

1.8. ГРАФІЧНІ МОЖЛИВОСТІ QBasic

1.8.1. Режими екрана

Існують два режими виведення інформації на екран - текстовий і графічний.

Текстовий режим дозволяє виводити 80 символів в одному рядку й містить на екрані 25 рядків (один - службовий). Такі параметри екрана встановлюються при ввімкненні комп'ютера й зберігаються при роботі QBasic.

В QBasic існують спеціальні графічні оператори для створення зображень. Вони вимагають перемикання в графічний режим роботи екрана оператором **SCREEN**.

Графічні режими характеризуються кількістю точок по вертикальній і горизонтальній висях екрана.

SCREEN *режим% [,[перемик_кольору][,активн_стор][,видима _стор]]]*

Вибір графічного режиму визначається параметром:

- *режим%*: 0-текстовий; 1,7,13 – *320*200*; 8 – *640*200*; 9 – *640*350*; 11,12 – *640*480*
- *перемик_кольору* – 0/1 – встановлює монохромний або кольоровий режим
- *активн_стор* – сторінка екрана, у яку здійснюється виведення тексту або графіки
- *видима_стор* - – сторінка екрана, відображувана на екрані в цей момент.

1.8.2. Кодування графічних зображень. Принципи подання зображень

Комп'ютерна графіка – розділ інформатики, що вивчає роботу на комп'ютері із графічними зображеннями (рисунками, кресленнями, фотографіями, відеокадрами та ін.)

Найпоширенішими є два принципи подання зображень: векторний і растровий.

Растрова графіка – засоби і методи комп'ютерної графіки, що використовують подання графічної інформації у вигляді сукупності кодів пікселів-складових зображення.

Піксель – найменший елемент зображення на екрані. Чим щільніше розташовані пікселі, тим краще виглядає зображення на екрані. Щільність пікселів виміряється як їх кількість на одиницю довжини. Найпоширеніша одиниця: **dpi** – кількість точок на дюйм (як правило, 72 або 96 dpi).

Растр – прямокутна сітка пікселів на екрані.

Розв'язувальна здатність екрана – розмір сітки растра (**M** x **N**- добуток числа точок по горизонталі на число точок по вертикалі.)

Відеоінформація – інформація про зображення, яка зберігається в пам'яті і відтворюється на екрані комп'ютера.

Відеопам'ять – оперативна пам'ять, що зберігає відеоінформацію під час її відтворення в зображенні на екрані.

Сторінка – частина відеопам'яті, що містить інформацію про один образ екрана (однієї “картинки” на екрані). У відеопам'яті можуть одночасно розміщатися кілька сторінок.

Графічний файл – файл, що зберігає інформацію про графічне зображення (.bmp, .jpeg)

Число кольорів, відтворених на екрані дисплея (**H**), можна підрахувати, знаючи число біт, що відводяться у відеопам'яті під кожний піксель (**n**).

$$H=2^n$$

n - бітова глибина (довжина коду пікселя)

n=1 - монохромний монітор, 2 кольори; n=24 - кольоровий монітор, 16777216 кольорів.

Все різноманіття фарб на екрані отримують шляхом змішування трьох базових кольорів: червоного, синього й зеленого.

Це змішування аналогічне змішуванню акварельних фарб на папері, але з однією відмінністю: колір акварельних фарб відтворюється у результаті відбиття падаючого на них світла; у той час, як колір на екрані формується в результаті випромінювання світла. Тому, коли ви змішуєте на папері три основні фарби, то одержуєте чорний колір, а для цих же кольорів максимальної яскравості на екрані виходить білий колір.

Кожний піксель на екрані складається із трьох близько розташованих елементів, що світяться цими кольорами. Кольорові дисплеї, що використовують такий принцип, називаються RGB - моніторами. Код кольору пікселя містить інформацію про частку кожного базового кольору.

Якщо всі три складові мають однакову інтенсивність (яскравість), то з їхніх сполучень можна одержати $2^3=8$ різних кольорів. У таблиці 5 наведене кодування 8-кольорової палітри за допомогою 3-розрядного двійкового коду. У ній наявність базового кольору позначено 1, а відсутність - 0.

Таблиця 5

Двійковий код 8-кольорової палітри			
к	**з**	**с**	**Колір**
0	0	0	Чорний
0	0	1	Синій
0	1	0	Зелений
0	1	1	Блакитний
1	0	0	Червоний
1	0	1	Рожевий
1	1	0	Коричневий
1	1	1	Білий

Більшу кількість кольорів можна отримати при роздільному управлінні інтенсивністю базових кольорів. Причому інтенсивність може мати більше двох рівнів, якщо для кодування кожного з базових кольорів виділяти більше одного біта.

Приклад. При використанні бітової глибини 8 біт/піксель кількість кольорів 2^8=256. Біти такого коду розподілені в такий спосіб:

ККК333СС - *ч*ервоний і зелений компоненти мають 2^3=8 рівнів яскравості, а синій – 4.

Відзначимо, що растрова графіка працює з реалістичними (фото) зображеннями.

Растрові файли потребують великого обсягу пам’яті (потрібен стиск при зберіганні).

При зміні розмірів, обертанні та інших перетвореннях рисунка відбувається його викривлення.

Векторна графіка – засоби і методи комп'ютерної графіки, що використовують подання графічної інформації у вигляді **графічних примітивів** - сукупності простих елементів: прямих ліній, дуг, кіл, прямокутників, зафарбувань та ін.

Положення й форма графічних примітивів задаються в системі графічних координат, пов'язаних з екраном. Початок координат розташований у верхньому лівому куті екрана.

Сітка пікселів збігається з координатною сіткою. Горизонтальна вісь X спрямована зліва направо, вертикальна вісь Y - зверху вниз.

Відрізок прямої лінії однозначно визначається вказівкою координат його кінців; коло - координатами центра і радіусом; багатокутник - координатами його кутів; зафарбована область - граничною лінією і кольором зафарбування та ін.

Графічні файли векторного типу мають невеликий обсяг.

Векторні зображення легко масштабуються без втрати якості.

Варто пам'ятати:

- при введенні зображень у комп'ютер за допомогою сканера формуються графічні файли растрового типу;
- при виведенні на екран монітора будь-якого зображення, у відеопам'яті формується інформація растрового типу;
- розходження в поданні графічної інформації в растровому і векторному форматах існує лише для графічних файлів.

Крім мов програмування графіка використовується в графічних редакторах - прикладних програмах, призначених для створення й обробки графічних зображень на екрані.

Приклади. Векторні редактори CorelDraw, AdobeIllustrator.

Приклади. Растрові редактори Paint, CorelPHOTO-PAINT, AdobePhotoshop.

1.8.3. Графічні примітиви QBasic

Для зображення графічних примітивів у QBasic використовуються відповідні оператори:

- **PSET**, **PRESET** - рисування точки;
- **LINE** - рисування відрізка;
- **CIRCLE** - рисування кола.

Існують три різних типи координат графічного екрана.

- абсолютні координати;
- координати на основі Точки Останнього Посилання, у скороченні — ТОП (***Last Point Referenced- LPR);***
- відносні координати.

Абсолютні координати

З огляду на систему координат екрана, просто вказується місце, у якому треба нарисувати точку, **наприклад**, PSET(100,120)

Значення в дужках (100 і 120) вказують на положення точки, що рисується, по осях **x** і **y** відповідно.

Точка Останнього Посилання

Координати точки, що була нарисована останньою, зберігаються в пам'яті комп'ютера. Ця точка і називається *Точкою Останнього Посилання* й часто використовується в графічних операторах.

Приклад. При рисуванні лінії за допомогою оператора LINE - (300,120) досить указати координати тільки однієї точки, і на екрані буде проведений відрізок від ТОП до зазначеної точки, що після цього стане ТОП.

Відразу після ввімкнення графічного режиму ТОП є точка в центрі екрана.

Відносні координати

Ці координати показують величину переміщення щодо положення ТОП. Щоб нарисувати нову точку, використовуючи відносні координати, буде потрібно ключове слово **STEP**.

Приклад. **PSET STEP (-5,8)**

При цьому з'явиться точка, положення якої буде ліворуч на 5 і нижче на 8 точок відносно ТОП. Інакше кажучи, якщо ТОП має координати (100,100), то даний оператор означає рисування точки з координатами (95,108).

Абсолютні координати повинні бути завжди додатними, а відносні можуть бути як додатними, так і від’ємними.

1.8.4. Оператори PSET і PRESET

Оператор **PSET** призначений для рисування точки на екрані шляхом зміни її кольору з фонового (чорного) на білий.

Оператор може мати такі форми:

PSET *(X, Y)* — абсолютна форма,
PSET STEP *(X, Y)* - відносна форма,
PRESET *(X,Y)* — абсолютна форма,
PRESET STEP *(X,Y)* - відносна форма,

де *X, Y* — абсолютні координати або зсув точки відносно ТОП.

Оператор **PRESET** призначений для зміни кольору відповідної точки на фоновий, тобто виконує дію, зворотну **PSET**.

1.8.5. Прямі лінії, відрізки, прямокутники

Оператор **LINE** призначений для рисування відрізка, що з'єднує дві довільні точки екрана.

Загальна форма запису:

LINE[(*X_початок, Y_початок***)]-(***X_кінець, Y_кінець***)[,[***колір***][,[***B/BF***], [,***стиль%***]]]**

де *X_початок, Y_початок* — координати початку відрізка (необов'язкові параметри);
X_кінець, Y_кінець — координати кінця відрізка (обов'язкові параметри);
колір – колір лінії;
B/BF – рисує прямокутник або зафарбований прямокутник замість лінії;
стиль% - чи будуть рисуватися точки растра.

Якщо координати початку відрізка не зазначені, то відрізок буде починатися в ТОП. В операторі **LINE** можна використати відносні координати початку і/або кінця відрізка.

Для рисування прямокутників можна вибрати більш простий шлях.

Приклад. **LINE (50,50)-(150,155) , ,B**

У випадку пропуску якого-небудь параметра або параметрів потрібно зберегти необхідну кількість розділовими комами.

Команда **DRAW** дозволяє:

- рисувати лінії;
- розфарбовувати області;
- обертати зображення;
- змінювати розміри (масштаб).

Загальний вид команди:

DRAW *текстова постійна*

текстова постійна - послідовність параметрів-кодів, які означають напрямок руху (команди). Після кожного параметра стоїть число, що вказує довжину лінії. Команда визначається однією або двома літерами, що задають конкретну дію.

Приклад. Ln - перемістити вліво P n, m - зафарбувати область.

а) переміщення

DRAW "M 20,20" - рисує лінію від ТОП до точки з координатами (20,20) (Аналог **Line -(20,20)).**

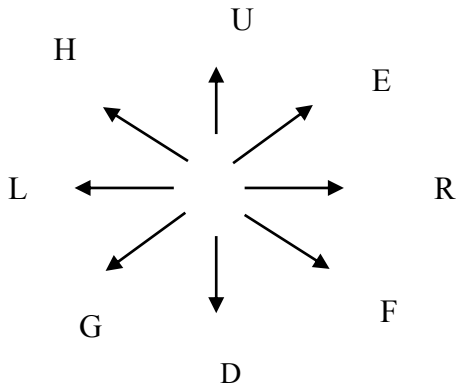
DRAW "M +25,-40" - рисує лінію від ТОП до точки на 25 точок праворуч і на 40 вище ТОП.

б) відносний рух

використовується для рисування ліній різних кольорів у вісьмох припустимих напрямках або переміщення в кожному із цих напрямків

Кожна з команд супроводжується цілочисловим аргументом, що вказує довжину лінії в точках

DRAW "R16"



Корисними також є такі параметри

- **B** - переміщення без рисування;
- **N** - переміщення без зміни ТОП;
- **C** - якщо він знаходиться в середині рядка символів разом із числовим значенням кольору, то колір ліній зміниться на зазначений.

Приклад

```
REM Рисування літери "B"
SCREEN 1
DRAW "BM+10, +0 R20 E10 U10 H5 E5 U10 H10 L20 D50"
DRAW "BM+10, -10 R10  U10 L10 D10"
DRAW "BM+0, -20 R10  U10 L10 D10"
```

в) обертання

Команда **An** обертає зображення на кут, кратний 90 градусам, де n=0,1,2,3

Команда **TAn** дозволяє повертати зображення на довільний кут від -360 до +360 (- означає поворот за годинниковою стрілкою; + проти годинникової)

Приклад

```
REM Рисування літери "B" з поворотом на 90 градусів
DRAW "A1"
```

г) масштабування

Команда **Sn** змінює розмір зображення, збільшуючи або зменшуючи його залежно від значення **n**.

Зображення масштабується в **n**/4 разів, тому команда "S12" збільшує лінійний розмір зображення в 3 рази, а команда "S2" - зменшує вдвічі (**n** може мати значення від 1 до 225).

Приклад

```
REM Рисування літери "B", збільшеної вдвічі
DRAW "S8"
```

д) колір в операторі DRAW

Колір може бути визначений за допомогою команди **Cn.** Зафарбовує замкнуту область.

Приклад

```
REM Рисування літери "B" рожевого кольору
DRAW "C2"
DRAW P колір, контур
```

1.8.6. Оператор CIRCLE

Дозволяє рисувати коло у будь-якому місці екрана:
CIRCLE (*X_центр, Y_центр*) , *радіус* — абсолютна форма
CIRCLE STEP (*X_центр, Y_центр*), *радіус* — відносна форма
де *X_центр, Y_центр* — координати або зсув центра кола; *радіус* — радіус кола.

Приклад.

```
REM Рисування кола
SCREEN 2
CLS
CIRCLE (100,100),25
```

Коло із центром у точці з координатами (100,100) має радіус 25 точок.

1.8.7. Використання кольору

Команда **COLOR** *A* [,*B,C*], де *A* - задає колір зображення, *B* - задає колір фону, *C* - задає колір границі.

Колір, що використовується як параметр в операторах **PSET**, **PRESET**, **LINE**, **CIRCLE**, впливає тільки на зображення, залишаючи тіло без змін.

У графічних операторах QBasic колір визначається в такий спосіб

PSET (*X, Y*) , *колір*

PRESET (*X,Y*) , *колір*

LINE (*X_початок, Y_початок*)- (*X_кінець, Y_кінець*) , *колір*

CIRCLE (X*_центр, Y__центр*) , *радіус, колір*, де *колір* - значення колірного параметра.

У режимі екрана, що задається оператором **SCREEN 2**, можливі тільки два кольори - чорний і білий, тому даний параметр марний.

Режим **SCREEN 1** підтримує 4 кольори, яким відповідають значення від 0 до 3.

Щоб "стерти" який-небудь елемент зображення без очищення всього екрана, можна просто перерисувати цей елемент кольором фона.

Команда **PAINT** використовується для розфарбовування замкнутих ділянок (не тільки прямокутних)

PAINT (*X,Y),D,C*

X,Y - координати будь-якої точки всередині замкнутої ділянки, що розфарбовують; *D* - колір розфарбовування; *C* - колір границі ділянки.

1.8.8. Дуга, еліпс і сектор

Щоб нарисувати дугу, еліпс або сектор кола, необхідно додати нові параметри в оператор **CIRCLE**, повна форма якого має такий вигляд:

CIRCLE (*X,Y), радіус, колір, початок, кінець, коефіцієнт*

- *X, Y* — координати центра кола;
- *радіус* — радіус кола;
- *колір* — його колір;
- *початок* — початкова точка дуги, задана в радіанах;
- *кінець* — кінцева точка дуги, задана в радіанах;
- *коефіцієнт* — відношення значень *Y*-радіуса і *X* - радіуса

Для рисування кола використовуються тільки параметри *X, Y* і *радіус*.

Для рисування дуги необхідно додати значення параметрів початкової й кінцевої точок. Дуга визначається кутом, що

вирізається з відповідного кола. Значення параметрів *початок* і *кінець* задаються в радіанах і повинні мати значення між 0 і 2π.

QBasic при рисуванні дуг веде відлік від початкової точки дуги до кінцевої проти годинникової стрілки.

Приклад

```
REM Рисування кола, дуги і сектора
SCREEN 2
CLS
CIRCLE (100,100), 30
CIRCLE (180,100), 30, 3, 1, 2
CIRCLE (260,100), 30, 3, -2, -1
```

Якщо одному з параметрів *(початок* або *кінець)* значення не присвоюється, то воно вважається рівним нулю.

При від'ємних значеннях цих параметрів QBasic з'єднує початкові кінцеві точки дуги із центром відповідного кола. Таким чином, на екрані створюється зображення сектора кола. Якщо від'ємним є значення тільки одного параметра, то й з'єднуватися із центром кола буде тільки одна точка дуги.

Для рисування еліпса потрібно задати *коефіцієнт* відношення радіусів по осях *Y* і *X*. Цей параметр визначає ступінь стиску еліпса й може мати будь-яке додатне значення.

Якщо параметр *коефіцієнт* опущений або дорівнює 1, ви одержуєте зображення кола. При від'ємному значенні параметра ви одержите повідомлення про помилку.

Приклад

```
REM Рисування еліпсів
SCREEN 2
CLS
CIRCLE (50,90), 30
CIRCLE (150,90),30, , , ,0.3
CIRCLE (250,90,30, , , ,1.5
```

1.8.9. Імітація руху на екрані

Для імітації руху зображення об'єкта на екрані необхідно виконати такий алгоритм:

1) нарисувати об'єкт у заданій точці;
2) знищити об'єкт, зафарбувавши його кольором фону;
3) змінити координати об'єкта;
4) перейти до п.1

Приклад

```
REM рух униз зеленого кола на чорному фоні
SCREEN 8
COLOR 2,1
Y=10: SY=4
```

```
50 CIRCLE (120,Y),40,2          'рисування кола
CIRCLE (120,Y),40,1             'стирання кола
Y=Y+SY                          ' зміна координати центра
IF Y<10 OR Y>190 THEN SY=-SY
GOTO 50
```

Варто організовувати штучні паузи між рисуванням і стиранням об'єктів для підтримки правильних зображень на екрані ("холості" цикли).

Оператори GET і PUT

З їхньою допомогою можна домогтися ефекту мультиплікації: створювати багаторазові копії зображення або рухати графічну картинку.

GET дає можливість зберегти будь-яку прямокутну область екрана в числовому масиві, а **PUT** відтворює зображення в довільному місці екрана.

Загальна форма запису

GET [STEP] *(x1!,y1!)* **- [STEP] (***x2!,y2***!), ***ім'я* **[(***індекс%***)]**
PUT[STEP] (*x1!,y1***!), *ім'я* [(***індекс%***)] [,***режим***]**

- **STEP** - задає відносну форму графічних координат;
- *x1!,y1!* - координати верхньої лівої точки зображення, що зберігається оператором GET або точка на екрані, починаючи з якої оператор PUT поміщає зображення;
- *x2!,y2!* - координати нижньої правої точки зображення, що зберігається;
- *ім'я_* - ім'я масиву, у якому зберігається зображення;
- *індекс%* - індекс елемента масиву, починаючи з якого зберігається зображення;
- *режим* - ключове слово, що визначає режим відтворення збереженого зображення (за замовчуванням - *XOR*). Може набувати таких значень:

AND - об'єднання збереженого зображення з існуючим;
OR - накладення збереженого зображення на існуюче;
PSET - рисування збереженого зображення, знищуючи існуюче;
PRESET - рисування збереженого зображення кольором тіла, знищуючи існуюче;
XOR - рисування збереженого зображення або знищення попереднього, зі збереженням і відновленням фона, відтворюючи ефект анімації.

2. ПРОЕКТУВАННЯ ПРОГРАМ ЗАСОБАМИ VISUAL BASIC 6.0

Середовище програмування QBasic використовується переважно для розроблення консольних додатків (тобто програм, які виконуються в середовищі MS-DOS). Виникає питання: чим відрізняється Windows та як писати для неї програми?

2.1. ВІДМІННІ РИСИ WINDOWS

Windows - це графічний інтерфейс користувача (Graphical User Interface, GUI). Іноді його ще називають "візуальний інтерфейс" або "графічне віконне середовище". Концепції, що дали початок цьому типу інтерфейса користувача, беруть свій початок у середині сімдесятих років. Після появи комп'ютера Macintosh, графічні інтерфейси користувача швидко поширилися, причому як у сфері персональних комп'ютерів, так і не персональних комп'ютерів. Зараз усі погоджуються, що графічний інтерфейс користувача є найважливішим "великим досягненням" у сфері персональних комп'ютерів.

Концепції й обґрунтування GUI

Всі графічні інтерфейси користувача уможливлюють використання графіки на растровому екрані дисплея. Графіка дає краще сприйняття дійсного стану речей на екрані, візуально багате середовище для передачі інформації й можливість WYSIWYG (What you see is what you get, що ви бачите, те й отримаєте) як для графіки, так і для форматованого друку документів тексту.

У перші дні свого існування дисплеї використовувалися винятково для відображення на екрані тексту, що користувач уводив із клавіатури. У графічному інтерфейсі користувача дисплей сам стає джерелом, звідки в машину вводиться інформація. Дисплей показує різні графічні об'єкти у вигляді картинок і конструкцій для введення інформації, такі як кнопки або смуги прокручування. Використовуючи клавіатуру або мишу, користувач може безпосередньо маніпулювати цими об'єктами на екрані. Графічні об'єкти можна перетягувати, кнопки можна натискати. Взаємодія між користувачем і програмою стає, таким чином, більше тісною. Замість послідовного введення інформації із клавіатури в програму і на дисплей, користувач взаємодіє з об'єктами безпосередньо на дисплеї.

Користувачам тепер не треба витрачати занадто багато часу на те, щоб навчитися користуватися комп'ютером і встановлювати нові програми. Windows сприяє цьому, оскільки всі програми для неї виглядають і сприймаються однаково.

Будь-яка програма для Windows має вікно - прямокутну область на екрані. Вікно ідентифікується заголовком. Більшість функцій програми запускається за допомогою меню. Занадто великий для екрана обсяг інформації може бути переглянутий за допомогою смуг прокручування. Деякі пункти меню викликають появу вікон діалогу, у які користувач уводить додаткову інформацію.

Після того, як ви навчилися працювати з однією програмою для Windows, вам буде легко навчитися працювати з іншою. Меню і вікна діалогу дають можливість користувачеві експериментувати з новою програмою й досліджувати її властивості.

Переваги багатозадачності

Під Windows будь-яка програма стає резидентною. Хоча резидентні програми не є багатозадачними програмами, вони дозволяють здійснювати швидке перемикання контексту. Таке перемикання контексту в принципі засноване на тих же концепціях, що й багатозадачність. Одночасно кілька програм Windows можуть мати виведення на екран і виконуватися. Кожна програма займає на екрані прямокутне вікно. Користувач може переміщати вікна по всьому екрана, змінювати їхній розмір, перемикатися між різними програмами й передавати дані від однієї програми до іншої.

Незалежність графічного інтерфейсу від устаткування

Windows - це графічний інтерфейс, і програми для Windows можуть повністю використовувати графіку й форматований текст як на дисплеї, так і на принтері. Графічний інтерфейс не тільки більш зручний для сприйняття, але він може також забезпечити користувачеві високоякісне відображення інформації.

У програм, написаних для Windows, немає прямого доступу до апаратної частини пристроїв відображення інформації, таких як екран і принтер. Замість цього Windows містить у собі мову графічного програмування, яка називається графічним інтерфейсом пристрою (Graphics Device Interface, GDI), що полегшує створення графіки та форматованого тексту. Windows абстрагується від конкретного пристрою відображення інформації. Програми, написані для Windows, будуть працювати з будь-яким типом дисплея і будь-яким типом принтера, для яких

є в наявності драйвер Windows. У програмі немає необхідності задавати тип обладнання, що використовується в системі.

Виклики функцій

Windows підтримує понад тисячу викликів функцій, які можна використовувати в додатках. Більшість програмістів, що пишуть програми для Windows, витрачають велику кількість часу на пошуки різних викликів функцій у друкованих першоджерелах або в системах контекстно-залежної підказки.

Кожна функція Windows має розгорнуте ім'я, написане буквами як верхнього, так і нижнього регістрів. У програмі для Windows ви використовуєте виклики функцій Windows приблизно так само, як використовуються бібліотечні функції. Основна відмінність у тому, що код бібліотечних функцій Basic зв'язується з кодом вашої програми, тоді як код функцій Windows залишається поза вашою програмою в бібліотеках, що підключають динамічно (DLL).

2.2. ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

При програмуванні для Windows ви фактично займаєтесь одним з видів об'єктно-орієнтованого програмування (Object Oriented Programming).

Об'єктно-орієнтоване програмування (ООП) – *це методологія програмування, заснована на поданні програми у вигляді сукупності об'єктів, кожний з яких є екземпляром певного класу, а класи утворюють ієрархію спадкування*.

Це найбільш очевидно для об'єкта, з яким ви у Windows будете здебільшого працювати, об'єкта, що дав Windows її назву, об'єкта, що відомий як "вікно".

Як уже згадувалося, вікна - це прямокутні області на екрані. Вікно одержує інформацію від клавіатури або миші користувача й виводить графічну інформацію на своїй поверхні. Вікно додатка звичайно містить заголовок (title bar), меню (menu), рамку (sizing border) і іноді смуги прокручування (scroll bars). Вікна діалогу - це додаткові вікна. Більш того, у вікні діалогу завжди є ще кілька вікон, які називаються "дочірніми" (child windows). Ці дочірні вікна мають вигляд кнопок (push buttons), перемикачів (radio buttons), прапорців (check boxes), полів текстового введення або редагування (text entry fields), списків (list boxes) і смуг прокручування (scroll bars).

Користувач розглядає вікна на екрані як об'єкти й безпосередньо взаємодіє із цими об'єктами, натискаючи кнопки й

перемикачі, пересуваючи бігунок на смугах прокручування. Досить цікаво, що положення програміста аналогічно положенню користувача. Вікно одержує від користувача інформацію у вигляді віконних "повідомлень". Крім цього, вікно обмінюється повідомленнями з іншими вікнами.

Об'єктом називають фрагмент програмного коду, який має певні властивості й методи.

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

Значення цих властивостей: наприклад, розмір тексту - кількісна характеристика; для властивості Visible, що встановлює видимість об'єкта, True або False - якісна характеристика.

Методи – це програмні процедури (або фрагменти коду), які виконують деяку обробку, пов'язану з об'єктом. Наприклад: після натискання на командну кнопку, виконуються певні розрахунки - для цього фрагмент програмного коду необхідно додати до тіла процедури командної кнопки.

Властивості і методи в ООП називаються **інтерфейсом об'єкта**.

Інтерфейс – це сукупність засобів, що забезпечують фізичну або логічну взаємодію пристроїв і програм обчислювальної системи або користувача з комп'ютером.

Принципи ООП**:**

1) абстракція;
2) інкапсуляція ;
3) ієрархія;
4) поліморфізм.

Абстракція:

- абстракція *виділяє істотні характеристики деякого об'єкта, що відрізняють його від всіх інших видів об'єктів;*
- вибір правильного набору абстракцій для заданої предметної області являє собою головне завдання ООП.

Інкапсуляція:

- інкапсуляція – *це процес відділення одне від одного елементів об'єкта, що визначають його призначення і повод жения; інкапсуляція служить для того, щоб ізолювати інтерфейс абстракції від їхньої реалізації;*
- під інкапсуляцією розуміється: приховування деталей реалізації; об'єднання даних і підпрограм їхньої обробки в рамках однієї синтаксичної структури мови програмування.

Ієрархія:

- ієрархія – *це впорядкування абстракцій, розташування їх по рівнях;*
- значне спрощення в розумінні складних завдань досягається за рахунок утворення з абстракцій ієрархічної структури;
- види ієрархій – агрегація, спадкування.

Спадкуванням називається здатність об'єкта зберігати атрибути класу або батьківського об’єкта.

Клас – це або шаблон, або проект, з якого надалі створюється об'єкт. Кожний об'єкт у цьому випадку є екземпляром класу, наприклад, готування печива: форма печива є класом, а печиво – об'єктом.

Поліморфізм:

- *здатність об'єкта приймати різні форми;*
- при цьому об'єкти можуть бути похідними інших об'єктів. У цьому випадку похідний об'єкт успадковує методи й властивості об'єкта батька;
- поліморфізм дозволяє додавати, видозмінювати, і навіть видаляти деякі особливості поводження похідного.

Основна кількість властивостей і методів об'єктів притаманні всім екземплярам класу, але деякі можуть бути змінені при необхідності, щоб характеризувати тільки даний екземпляр.

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

1) створення екранної форми (з об'єктами і їхніми властивостями);
2) написання програми (програмного коду);
3) відлагодження програми, тобто усунення в ній логічних помилок (після цього створюється програмний проект);
4) перетворення проекту у Windows- додаток.

Екранна форма – графічне подання вікна Windows-додатка разом з вмістом цього вікна.

Вміст цього вікна включає:

1) перелік властивостей цього вікна з їхніми значеннями;
2) перелік об'єктів, що перебувають у цьому вікні;
3) властивості цих об'єктів з їхніми значеннями.

Екранна форма зберігається в окремому файлі, який має розширення **frm.**

Програмний модуль - програмний код, що зберігається в окремому файлі з розширенням **bas.**

Переваги об’єктно-орієнтованого підходу:

- можливість модифікації коду окремих частин програми незалежно від інших;
- підвищення надійності програм;
- можливість повторного використання коду компонентів програми.

Недоліки:

- вимагає додаткових витрат пам'яті;
- приводить до зниження швидкості виконання додатків.

2.3. АРХІТЕКТУРА, КЕРОВАНА ПОДІЯМИ

У Windows, коли користувач змінює розмір вікна, програмі відправляється повідомлення про новий розмір вікна. Після цього програма може змінити розміри свого вікна на нові.
Коли говориться: "Windows посилає програмі повідомлення," - мається на увазі, що Windows викликає функцію усередині програми. Параметри цієї функції описують параметри повідомлення. Ця функція називається віконною процедурою (window procedure).

Розглянемо як відбувається процес розроблення Windows-додатка в системі програмування Visual Basic (VB).

Що ж залишилось від мови QBasic і що нового з’явилося в VB**?**

Найбільш характерним підходом було й залишається навчання за принципом "що знаємо, те й викладаємо", не піклуючись про питання, "а навіщо це потрібно". Сьогодні програмування є частиною проблеми комп'ютерної грамотності, і методика його викладання повинна кардинально змінитися. Технологія розроблення програм: візуальне програмування, подійна логіка програми, компонентна технологія, використання макрозасобів і т.ін., повинна докорінно відбитися на методиці навчання. А що ж робиться зараз? Частіше взагалі просто вивчається теорія мов. Але така постановка питання нагадує відомий анекдот про басейн, що обіцяють наповнити водою, коли все навчаться плавати.

Класична схема навчання програмуванню раніше була така: спочатку теоретичне вивчення синтаксису деякої мови, а вже потім виконання деяких практичних завдань.

Сьогодні ж для написання перших програм для Windows взагалі не потрібно знати про особливості мови — потрібно

розуміти загальну логіку розроблення додатка і вміти працювати в середовищі інструмента. І тільки після цього можна переходити до вивчення мови для "вишуканого програмування". Який засіб розроблення придатний для вирішення подібного завдання? Відповідь - Visual Basic.

По-перше, це дозволить максимально абстрагуватися від мовних проблем.

По-друге, ознайомитися із середовищем програмування, характерним для всіх засобів розроблення Microsoft . Освоєння ж мови реально можливе тільки в ході практичної роботи й самонавчання.

У 1991 році під гаслом "тепер і починаючі програмісти можуть легко створювати додатки для Windows" з'явилася перша версія нового інструментального засобу Microsoft Visual Basic. У той момент Microsoft досить скромно оцінювала можливості цієї системи, орієнтуючи її насамперед на категорію починаючих і непрофесійних програмістів. Основне завдання тоді полягало в тому, щоб випустити на ринок простий і зручний інструмент розробки в новому середовищі Windows, програмування в якій являло собою проблему навіть для досвідчених фахівців. Тому VB версії 1.0 був схожий скоріше на діючий макет майбутнього середовища розробки, ніж на робочий інструмент.

Однак уже тоді принципове нововведення VB полягало в реалізації ідей подійно-керованого та візуального програмування в середовищі Windows, які радикально відрізнялися від класичних схем розроблення програм. За загальним визнанням, VB став родоначальником нового покоління інструментів, які сьогодні називаються засобами швидкого розроблення програм (Rapid Application Development, RAD). Зараз ця ідеологія вже звична, але тоді вона здавалася зовсім новою, і це створювало серйозні проблеми (у тому числі психологічного плану) для програмістів "старих часів".

Проте число VB-користувачів росло, причому багато в чому за рахунок величезної популярності її попередника - QuickBasic. При цьому VB швидко "мужнів" як у результаті розвитку середовища програмування, так і за рахунок включення в нього професійних елементів мови і проблемно-орієнтованих засобів.

На початку 90-х років намітилася виразна тенденція включати в додатки засоби внутрішнього програмування, які повинні були вирішувати завдання настроювання й адаптації цих пакетів для конкретних умов їхнього засТОПування.

Наприкінці 1993 р. Microsoft оголосила про намір створити на основі VB нову універсальну систему програмування для

прикладних програм, що одержала назвуVisual Basic for Applications (VB для додатків), або VBA. Природно, реалізацію цього проекту вона почала із власних офісних пакетів.

До складу MS Office 2000 увійшла версія VBA 6.0. Тепер вона використовується вже в шести програмах - Word, Excel, PowerPoint, Access, Outlook, Frontpage. Тому в останні три роки Microsoft представляє свій пакет MS Office не просто як набір прикладних програм, а як комплексну платформу для створення бізнес-додатків, що вирішують широке коло спеціалізованих завдань користувачів. Крім того, Microsoft оголосила про можливості ліцензування VBA для того, щоб зробити це середовище фактичним стандартом для керування програмувальними додатками.

На сьогоднішній день підсумки загального навчання програмуванню студентів технічних спеціальностей навряд чи можна назвати успішними. На практиці майже завжди виходило так, що той, хто дійсно хотів писати програми, опановував цю технологію самостійно. Абсолютна ж більшість студентів забували про програмування незабаром після здачі заліку.

До цього потрібно додати відверто слабку методичну основу викладання (мова йде про підготовку не професіоналів з обчислювальної техніки, а саме фахівців інших предметних галузей), що зводилося не стільки до вивчення програмування на прикладі конкретної мови, скільки до освоєння синтаксису мови без усякого зв'язку з технологією побудови алгоритмів і практичною реалізацією програм.

За останні кілька років комп'ютери стали істотно доступніше, але ситуація з вивченням програмування практично не покращилася. Одна із причин цього - принциповий сумнів у тому, що його взагалі потрібно освоювати. Дійсно, навіщо вивчати програмування, якщо можна скористатися численними готовими додатками? Досить просто навчитися "комп'ютерної грамоти" - вміти водити мишею й натискати на кнопки.

Можливо, така постановка питання й має раціональне зерно, але все-таки вона є досить обмеженою:

- по-перше, програмування допомагає краще формулювати логічні рішення практично будь-якої задачі (зовсім не обов'язково чисто обчислювальної);
- по-друге, саме розширення функціональності готових програм з неминучістю вимагає їх більш тонкого настроювання та адаптації до потреб конкретного користувача. Використання методів програмування різко розширює можливості вирішення

цього завдання. Причому залучати професійних розробників часто просто не має сенсу;

- по-третє, за десять років радикально змінилися інструменти розробки й технологія їхнього освоєння. Ці засоби стали істотно більш зрозумілими навіть на інтуїтивному рівні. Крім того, змінилася сама схема практичної роботи: якщо раніше потрібно було спочатку вивчити теорію програмування, щоб написати навіть простеньку програмку, то зараз серйозне опанування програмування починається звичайно після написання корисного додатка.

Зокрема, програмування з використанням офісних додатків відкриває унікальні можливості опанування технології розроблення для "звичайного" користувача. Буквально з перших кроків ви досягнете позитивних результатів і потім зможете поетапно збагачувати свої знання й навички програмування. Однак потрібно мати на увазі одну важливу істину: для переходу до серйозного професійного розроблення тільки досвіду буде недостатньо - у якийсь момент буде потрібно вивчення теорії.

У цьому розділі автори не мали за мету навести весь базовий «арсенал» VB. Він в основному властивий всім мовам програмування (наприклад, змінні й команди циклу, опис базових типів даних і т.ін.).

Написані на VB програми досить близькі за стилем до традиційного програмування ранньої епохи Basic (якщо не брати до уваги деяких дивних, але необхідних синтаксичних конструкцій).

2.4. ОСНОВНІ ЕЛЕМЕНТИ ПРОГРАМУВАННЯ VB6

Для керування ходом виконання програм у VB6 можуть використовуватися (і використовуються) засоби, детально описані в розд. 1, присвяченому QBasic. Це оператори вибору, циклів, переходу, опису та звернення до процедур, обробки символів, побудови графічних примітивів і т.д.

Розглянемо деякі типи даних та засоби створення програм VB6, які відрізняються від застосованих в QBasic або не були описані в попередньому розділі.

2.4.1. Типи даних

Тип даних – це термін, що відноситься до визначених видів даних, що VB6 зберігає і якими може маніпулювати.

Ви можете застосовувати такі **типи даних** (табл. 6):

- числовий;
- символьний (String);
- типу дата (Date);
- байтовий (Byte);
- логічний (Boolean);
- довільний (Variant);
- об'єктний (Object).

Для зберігання чисел використовуються такі типи:

- ціле число: Integer;
- довге ціле число: Long;
- десяткове число звичайної точності: Single;
- десяткове число подвійної точності: Double;
- десяткове довге число: Currency;
- однобайтове ціле число:Byte.

Таблиця 6

Тип даних	О'бєм займаної пам'яті (байт)	Діапазон значень	Префікс	Суфікс	Приклад
1	2	3	4	5	6
Цілі числа					
Byte (однобайтне ціле число)	1	Позитивне число від 0 до 255	byt		bytImage
Integer (коротке ціле число)	2	Від -32768 до 32767	int	%	intQuantity
Long (довге ціле число)	4	Від -2147483648 до 2 147483648	lng	&	lngTotal
Boolean (логічне значення)	2	True (ІСТИНА) або False (ХИБНІСТЬ)	bln		blnSuccess
Числа з плаваючою крапкою					
Single (речовинне число із плаваючою крапкою (нормальне, одинарне))	4	Від'ємні числа: від -3.4E+38 до -1.4E-45 Додатні числа: від 1.4E-45 до 3.4E+38	sng	!	sngLength
Double (речовинне число із плаваючою комою подвійної точності (довге))	8	Від'ємні числа: від -1.8D+308 до -4.9D-324 Додатні числа: від 4.9D-324 до 1.8D+308	dbl	#	dblSum

Продовження таблиці 6

1	2	3	4	5	6
String (рядок змінної довжини)	**10 байт + довжина рядка (1 байт на кожний символ)**	**Від 0 до 2 мільярдів символів**	**str**	**$**	**strLastname**
String *довжина (рядок фіксованої (постійної) довжини))	**довжина рядка (1 байт на кожний символ**	**Від 1 до ~65400**		**$**	
Об'єктні типи					
Object (посилання на об'єкт)	**4**				
Не визначені типи					
Variant (числові типи)	**16**	**Довільне числове значення**	**vnt**		**vntValue**
Variant (символьні типи)	**22 байта+ довжина рядка**	**Довільне символьне значення**	**vnt**		
Інші типи					
Currency (грошова величина-число з фіксованою крапкою)	**8**	**Ціла частина числа до 15 цифр, дробова – до 4 Від -922337203685477. 5808 до922337203685477.5807**	**cur**	**@**	**curPrice**
Date (дата/час)	**8**	**Діапазон дат від 1 січня 100 року до 31 грудня 9999 року. Діапазон часу від 00:00:00 до 23:59:59.**	**Dtm**		**dtmFinish**

Тип **Byte** – це найменший із трьох цілих типів даних VB6, призначений для збереження цілих чисел від 0 до 255. Зберігати негативні числа в типі Byte не можна. Його використання дозволяє сильно заощаджувати оперативну пам'ять і зменшувати розмір масивів. Звичайно типи Byte використовуються для збереження двійкових даних (графічних, звукових файлів і так далі).

Логічний або булевий тип даних VB6 називають також типом **Boolean.** Boolean – тип даних може зберігати тільки два значення: True або False, ІСТИНА або ХИБНІСТЬ (булеві значення). Його використання замість цілочислових змінних є хорошим стилем програмування. Звичайно, VB6-програма (як і будь-які інші) «приймає» вирішення, перевіряючи, чи є

виконаними різні умови, тобто в операціях порівняння даних. Якщо відображається тип Boolean на екрані, VB6 автоматично перетворить його в рядок, що містить або слово True, або False.

Для зберігання посилань на об'єкти використовується тип даних **Object.** Він зберігає адресу об'єкта в пам'яті. Кожна змінна даного типу вимагає 4 байти.

Тип даних **Variant** – це особливий тип даних, що може зберігати будь-які типи, наведені в табл.1, за винятком типу Object. VB6 використовує тип Variant для всіх змінних, для яких тип не визначений заздалегідь**.**

Дані типу Variant набувають характеристик визначеного типу, що зберігаються в даний момент. Наприклад, якщо дані типу Variant містять символьні дані, Variant набуває характеристик типу String (байт/символ). Якщо дані типу Variant містять числові дані, Variant набуває характеристик якого–небудь числового типу, звичайно – Double, хоча типи Variant можуть також мати характеристики типів Integer, Long, Single або Currency.

VB6 використовує для даних типу Variant найбільш компактне подання, можливе для конкретних значень, що знаходяться в даних.

Типи Variant вимагають більшого обсягу пам 'яті, ніж будь-який інший тип даних, за винятком великих рядків. Крім того, математичні операції та операції порівняння над даними типу Variant виконуються повільніше, ніж подібні операції над даними будь-якого іншого типу.

Змінну типу Variant можна використати для зберігання всіх типів даних і виконувати операції, не піклуючись про тип даних, що знаходиться в них. Необхідно тільки пам'ятати про два винятки. По-перше, виконувати арифметичні операції або функції над змінною типу Variant можна тільки в тому випадку, якщо вона містить числове значення. По-друге, конкатенацію рядків варто здійснювати за допомогою оператора «&», замість оператора «+».

Можна використовувати вбудовані функції для перевірки типу даних, що зберігаються в змінній типу Variant. Вони дозволяють легко перевірити, чи правильно користувач вводить інформацію. Використання такого типу даних, як Variant, уповільнює роботу програми, тому що потрібний час і ресурси для операцій перетворення типів. Крім того, багато програмістів розуміють, що використання автоматичних перетворень типів даних призводить до неакуратного виду програм. Єдина причина

використання Variant полягає в можливих помилках при перетворенні типів безпосередньо програмістом.

Тип **Currency –** це число з фіксованою крапкою**,** тобто десяткова крапка завжди знаходиться в тому самому місці– праворуч від десяткової крапки завжди стоїть чотири цифри. Він створений для того, щоб уникнути помилок при перетворенні десяткових чисел у двійкову форму й навпаки (неможливо 1/10 представити як суму 1/2,1/4, 1/8, 1/16 і т.п.). Використовується тип Currency для збереження чисел, коли точність вкрай важлива, що буває при «грошових» обчисленнях. Тип Currency вимагає 8 байтів пам'яті і може зберігати числа від –922337203685477,5808 до 922337203685477,5807. Математичні операції над числами типу Currency мають невеликі або зовсім не мають помилок округлення, тому вони точніші, ніж операції над числами з плаваючою крапкою. Помилки округлення з числами типу Currency звичайно виникають тільки тоді, коли виконується множення чи ділення числа типу Currency на значення іншого числового типу. Подібно обробці всіх інших чисельних типів даних VB6 автоматично перетворить значення Currency у текст, коли вони виводяться на екран. (Для тих, хто цікавиться: даний тип використовує цілі числа з 19 розрядів, які потім діляться на 10000. Це дозволяє організувати 15 знаків до коми і 4 після неї.) Усередині даного діапазону обчислення будуть точними.

Тип даних **Date** використовується для зберігання дати і часу. Змінна цього типу вимагає 8 байтів оперативної пам'яті. Date дозволяє зберігати значення часу і дати в проміжку від напівночі 1 січня 100 року до напівночі 31 грудня 9999 року. Такі значення в тексті програм позначаються символами «#».

Приклад. **NewYear = #January 1, 2000# .**

Якщо вводиться тільки значення дати, Visual Basic вважає, що час рівняється 00:00.

Іноді, при використанні деяких операторів, а також просто для власних потреб необхідно створювати власні типи даних. Часто їх називають структурами. За своєю суттю структура – це як би одновимірний масив, що ми поміщаємо в одну змінну. Але в нього можуть входити дані різних типів(див.п.1.2.4.3).

Створення власного типу даних здійснюється за допомогою інструкції Type, що використовується в секції General коду форми.

2.4.2. Змінні

Імена змінних

Для того щоб зробити змінні більш наочними і простими для читання, рекомендується давати їм імена, що мають певне смислове значення.

- ім'я може містити не більше 255 символів;
- імена не можуть містити пробіли, крапку або будь-який інший символ, що VB6 використовує для позначення математичних операцій і операцій порівняння (=, +, - і т. п);
- ім'я повинне бути унікальним у рамках його області дії, тобто повинне бути унікальним у межах процедури або модуля,у якому оголошено цю змінну (області видимості).

Зауваження.

У найменуваннях змінних рекомендується використовувати префікси, що відбивають тип змінної та область її дії. При такому позначенні змінних підвищується читабельність програми і знижується кількість помилок програмування.

Оголошення типу змінної

Усі змінні, які VB6 створює неявним оголошенням змінної, мають тип даних Variant.

Краще використати явне оголошення змінних, тому рекомендується встановити такий режим трансляції програми, при якому допускається тільки явне оголошення змінних. Для цього необхідно в початок модуля вставити оператор **Option Explicit** (Явне оголошення). Для автоматичного додавання в усі модулі даного оператора у вікні програми VB6 необхідно виконати команду **Options** (Параметри) меню **Tools** (Сервіс). Відкриється діалогове вікно **Options**, на вкладці **Editor** якого потрібно встановити прапорець **Require Variable Declaration.**

Явне оголошення змінних має такі переваги:

- прискорюється виконання коду. VB6 створює всі оголошені явно змінні в модулі або процедурі перед виконанням коду процедури. Швидкість виконання коду збільшується;
- зменшується кількість помилок у результаті правильного написання імені змінної;
- код стає більш читабельним і зрозумілим. Бачачи всі оголошення змінних на початку модуля або процедури, користувач може легко визначити, які змінні використовуються в цьому модулі або процедурі;

• за допомогою явного оголошення змінних можна нормалізувати виділення великих букв в імені змінної. Якщо явно повідомляється про змінну, VB6 завжди змінює великі букви в імені змінної в операторі відповідно до імені в оголошенні змінної.

Оголошення типу операторами Dim, Private, Static, Public, які записуються на початку тексту програмного коду, визначають область дії змінної і мають такий синтаксис:

Dim *ім'я_змінної* [**As** *Тип_змінної*] [, *ім'я_змінної*: [**As** *Тип_ змінної*]]….
Private *ім'я_змінної* [**As** *Тип_змінної*] [, *ім'я_змінної*: [**As** *Тип_ змінної*]]….
Static *ім'я_змінної* [**As** *Тип_змінної*] [, *ім'я_змінної*: [**As** *Тип_ змінної*]]….
Public *ім'я_змінної* [**As** *Тип_змінної*] [, *ім'я_змінної*: [**As** *Тип_ змінної*]]….

Область дії змінних

При виконанні програми принципове значення має область дії змінних. Спроба використання змінних, які не діють у даному місці програми, призводить до помилки програмування або ж до неоднозначності результатів. В VB6 можна застосовувати глобальні та локальні змінні. Глобальні змінні доступні з будь-якої частини програми. Для локальних змінних можна задавати область дії в рамках усього модуля або окремої процедури.

Присвоюючи імена змінним з урахуванням області їх дії, дотримуйтеся формату, наведеного в табл.7:

Таблиця 7

Область дії змінної	**Префікс**	**Приклад**
Глобальна	**g**	**gdtmFinish**
Локальна всередині модуля	**m**	**msngLength**
Локальная всередині процедури	**Немає префікса**	**strLastname**

Для створення змінної, котру ви хочете визначити як глобальну, у розділ **General Declarations** головного модуля додатка помістіть оператор **Public**.

Для оголошення змінної, локальної усередині модуля або форми, використайте оператор **Private** або **Dim** у розділі **General Declarations** модуля або форми. У цьому випадку оголошена змінна буде доступна для всіх процедур, що входять у форму або модуль, але в той же час недоступною в процедурах інших модулів і форм.

Змінні, локальні на рівні процедури, створюються операторами **Dim** або **Static** усередині процедури.

2.4.3.Константи

Числові і символьні константи розділяються на дві великі групи:

- символічні або визначені користувачем константи, що оголошуються оператором **Const**;
- внутрішні або системні, що надаються додатками або елементами керування. Константи перераховані в бібліотеках об'єктів VB6 у вікні **Object Browser** (Перегляд об'єктів). Константи також визначені в об'єктній бібліотеці для кожного елемента керування **ActiveX**.

Як і у випадку зі змінними, можна оголошувати іменовані константи в процедурах або в області оголошень на початку модуля. Константа, що повідомляється в процедурі, має область дії процедурного рівня, а константа, що повідомляється в області оголошень модуля, – область дії модульного рівня.

Оскільки однією з головних цілей використання іменованої константи є запобігання повторення або дублювання значень літеральних констант у процедурах, як правило, буває необхідно, щоб іменовані константи були доступні всім процедурам у модулі. Тому варто розташовувати оголошення констант на модульному рівні, щоб у них була найбільша область дії .

VB6 містить величезну кількість вбудованих (системних) констант практично для всіх можливих випадків: кольору, клавіші, повідомлення і т.п. Вбудовані константи мають префікс **VB6.** Для пошуку констант певної категорії скористайтеся браузером об'єктів, що відкривається при натисканні кнопки Object Browser на стандартній панелі інструментів.

Оголошення констант

Оголошення констант багато в чому аналогічно оголошенню змінних. Константи можна повідомляти на рівні модуля або процедури. Область їхньої дії при цьому визначається тими ж правилами, що й для змінних.

Для оголошення константи на рівні процедури використовується оператор **Const**, що має такий синтаксис:

Const *ім'я_константи* **[As** *тип_даних***] =** *вираз*

При оголошенні константи на рівні модуля можна додатково вказати область її дії. У цьому випадку оператор **Const** має такий синтаксис:

[Public / Private] Const *ім'я_константи* **[As** *тип_даних***] =** *вираз*

2.4.4. Масиви

Масиви змінної розмірності (динамічні)

У випадку, коли розмір масиву заздалегідь невідомий, VB6 (до речі, як і QBasic) дозволяє використати динамічні масиви, розміри яких можна змінювати під час виконання. ЗасТОПування динамічних масивів дозволяє ефективно управляти пам'яттю, виділяючи пам'ять під великий масив лише на той час, коли цей масив використовується, а потім звільняючи її.

Оголошення динамічного масиву

Створення динамічного масиву здійснюється в такий спосіб:

- оголошується масив за допомогою ключових слів, використовуваних при створенні масиву фіксованого розміру. Список вимірностей масиву залишається порожнім. При оголошенні глобального масиву необхідно вибрати ключове слово **Public**, при оголошенні масиву на рівні модуля - **Dim**, при оголошенні масиву в процедурі - **Dim** або **Static**.

Синтаксис для динамічного масиву має вигляд

ReDim [Preserve] *Ім'я змінної* **(***Індекси***) [As** *Тип***] [,** *Ім 'я змінної* **(***Індекси* **)**
[As *Тип***]]**

Preserve - ключове слово, що використовується для збереження даних в існуючому масиві при зміні значення його розміру.

На відміну від масивів статичних розмірів, коли звертатися до елементів можна відразу після його оголошення, до елементів динамічного масиву відразу звертатися не можна, тому що вони ще не ініціалізовані;

- за допомогою виконуваного оператора **ReDim** вказується новий розмір масиву у вигляді числа або виразу.

При виконанні оператора **ReDim** дані, розміщені в масиві раніше, втрачаються. Це зручно в тому випадку, якщо дані вам більше не потрібні, і ви хочете перевизначити розмір масиву та підготувати його для розміщення нових даних. Якщо ви хочете змінити розмір масиву, не втративши при цьому даних, то необхідно скористатися оператором **ReDim** із ключовим словом **Preserve**.

Приклад

```
Dim myLong As Long
Dim myArray() As Long ' оголошуємо масив
ReDim myArray (2) ' одна вимірність [0,1,2]
myArray (1) = 234 ' присвоюємо другому елементу число 234
myLong = myArray (1) ' зберігаємо його в змінній myLong
ReDim myArray (3) ' знову міняємо тепер [0,1,2,3]
myLong = myArray (1) ' знову намагаємося зберегти другий елемент
```

В останньому рядку, змінній myLong присвоїться 0 замість 234! Це відбувається тому, що оператор **ReDim** заново ініціалізує (скидає) всі елементи масиву до значення за замовчуванням (як пам'ятаєте, для чисел - це 0, для рядків ""). Але як же бути, якщо ми хочемо змінити розміри масиву, зберігши всі старі елементи? Для цього потрібно після оператора **ReDim** поставити слово **Preserve**.

Приклад

```
ReDim Preserve myArray (3) ' зберігаємо старі елементи
myLong = myArray (1) ' усе в порядку
```

Зауваження. *Використання оператора* ***ReDim*** *із ключовим словом* ***Presterve*** *дозволяє змінювати тільки верхню границю останньої вимірності багатовимірних масивів.*

Масиви можуть зберігатися в змінних типу Variant. Іноді це буває зручним. У деяких випадках без цього просто не обійтися! (Наприклад, коли ви хочете, щоб ваша функція повертала масив).

Щоб зберегти який-небудь масив у змінній типу Variant, необхідно просто присвоїти цієй змінній потрібний масив:

Приклад

```
Dim myVariantArray ' змінна Variant за замовчуванням
myVarianrArray = chessTable
```

Зверніть увагу, ніякі індекси вказувати не потрібно!
Тепер можна використати копію як звичайний масив.

Приклад.

```
myVarianrArray (0) = "Це копія"
```

При роботі з масивами можна використовувати функції і процедури, наведені нижче.

Функція **Array** є зручним способом визначення одновимірних масивів, що перетворює список елементів, розділених комами, у вектор з цих значень, присвоюючи їх змінній тип Variant.

- **Array(***список_аргументів***)**

список_аргументів – це розділений комами список значень, які надано елементам масиву.

Приклад

```
Dim A As Variant
A = Array(10, 20, 30)
B = A(2)
```

Приклад

```
Dim День A s Variant
День = Array("Пн", "Вт", "Ср ", "Чт", "Пт")
```

- **IsArray (***ім'я_ змінної* **)**

Повертає True, якщо змінна містить масив; в іншому випадкуповертається False. Функцію IsArray використовують для перевірки значень змінних типу Variant, що містять масиви;

- **Erase** *список_ масивів*

Повторно ініціалізує елементи масивів фіксованої довжини і звільняє пам'ять, відведену для динамічного масиву.

Список_ масивів складається з імен масивів, які розділені комою.

Інструкція **Erase** встановлює елементи масивів фіксованої довжини таким чином :

- масив чисел або рядків фіксованої довжини - присвоює кожному елементу значення 0;
- масив рядків змінної довжини присвоює кожному елементу значення пустого рядка;
- масив типу Variant присвоює кожному елементу значення Empty.

2.4.5. Перерахування

Перерахування - це список констант. Перед використанням такого списку його необхідно визначити в програмі.

Приклад. Розглянемо перерахування оцінок, що отримали студенти:

```
Enum Ocenka
  Neud = 3
  Horosho = 4
  Otlichno = 5
End Enum
```

Присвоювати значення константам усередині Enum не обов'язково. Якщо цього не зробити, то константи будуть набувати значень 0,1,2... і т.д.

Тепер можна оголосити змінну типу Ocenka:

```
Dim oc1 As Ocenka
```

І якщо ви тепер спробуєте присвоїти такій змінній значення – VB6 видасть список (Neud, Horosho і Otlichno), з якого ви зможете вибрати потрібне значення.

2.4.6. Характерні мовні конструкції

Розглянемо застосування деяких мовних конструкцій в VB6.

Коментарі рядків

Прокоментувати ті або інші програмні рядки в VB6 можна з використанням відомих з QBasic **'** або **Rem**

Методи введення-виведення

Завдяки графічному інтерфейсу користувача, який розробляється у VB6 на самому початку проектування, введення-виведення даних значно спростилося. Детальніше на цьому ми зупинимося далі.

Розміщення оператора на декількох рядках

У тому випадку, коли оператор має велику довжину, його можна розбити на кілька рядків, використовуючи символи продовження рядка: пробіл, за яким розташований символ підкреслення (_).

Приклад. Помістимо на двох рядках оператор, що поєднує прізвище, ім'я та по батькові:

```
strName = strLastname & strFirstname & strSecondname
```

Маємо таке:

```
strName = strLastname _
& strFirstname & strSecondname
```

Керуючі конструкції

В VB6, як і у всіх мовах програмування, існують конструкції, призначені для керування порядком виконання команд.

Конструкція IF THEN ELSE використовується в тому випадку, коли необхідно, щоб група операторів виконувалася при дотриманні певних умов. Конструкція SELECT CASE дозволяє на підставі аналізу значення заданого виразу виконувати ті або інші дії. Синтаксис цих конструкцій описаний у розд. 1. Відзначимо, що складні умовні вирази можна попередньо обчислити й зберігати в логічних змінних типу Boolean.

Для реалізації повторюваних дій, крім описаних у розд. 1 For...Next і Do. . . Loop застОПовується конструкція:

FOR EACH...NEXT

Цей цикл схожий на цикл For.. .Next, але використовується для обробки всіх елементів деякого набору об'єктів або масиву. Він особливо зручний в тому випадку, коли кількість оброблюваних елементів невідома.

Синтаксис конструкції:

FOR EACH *елемент* **IN** *група*
конструкції
NEXT *елемент*

Приклад

```
Dim objControl As Control
For Each objControl In Controls
objControl.Caption = "Test " & objControl.Caption
Next objControl
```

Зауваження. *При використанні конструкції* ***For Each...Next*** *необхідно мати на увазі, що для набору об'єктів параметр* ***елемент*** *може бути тільки змінною типу Variant, загальною змінною типу Object або об'єктом, перерахованим в Object Browser. Для масивів параметр елемент може бути тільки змінною типу Variant.*

Присвоювання масивів

VB6 і версії вище дають можливість проводити операції присвоювання з масивами так само, як зі змінними. Тепер немає необхідності створювати цикл For...Next для присвоювання одного масиву іншому по кожному елементу. Досить написати такий оператор:

NewMassive=OldMassive

і вміст масиву **OldMassive** присвоїться масиву **NewMassive**.

Однак при цьому необхідно враховувати, що для виключення помилок при такому присвоєнні бажано дотримувати однакової розмірності і типу масивів. Хоча при присвоєнні динамічного масиву динамічному масиву, масив у лівій частині оператора змінюється, як би підбудовується під оператор у правій частині. Однак при роботі зі статичними масивами можлива помилка компіляції. Крім того, при присвоєнні, наприклад масиву типу Long типу Integer може виникнути помилка переповнення (Owerflow).

У програмі операція присвоєння може виглядати приблизно так (на формі повинні бути кнопка Command1 і текстбокс Text1).

Приклад

```
Option Explicit
Dim OldMassive() As Long
Dim NewMassive() As Long
Private Sub Command1_Click()
Dim x As Long
For x = 0 To 999 'просте заповнення масиву цифрами
ReDim Preserve OldMassive(x)
```

```
OldMassive(x) = x
Next x
NewMassive = OldMassive 'присвоєння масивів
For x = 0 To UBound(NewMassive) ' зчитування нового масиву в Text1
Text1.Text = Text1.Text & NewMassive(x) & VB6CrLf
Next x
End Sub
```

2.5. СЕРЕДОВИЩЕ VISUAL BASIC 6.0

При першому запуску **VB6** запускається майстер **Project Wizard**, і на екрані з'являється діалогове вікно **New Project** (Новий проект) (рис. 11).

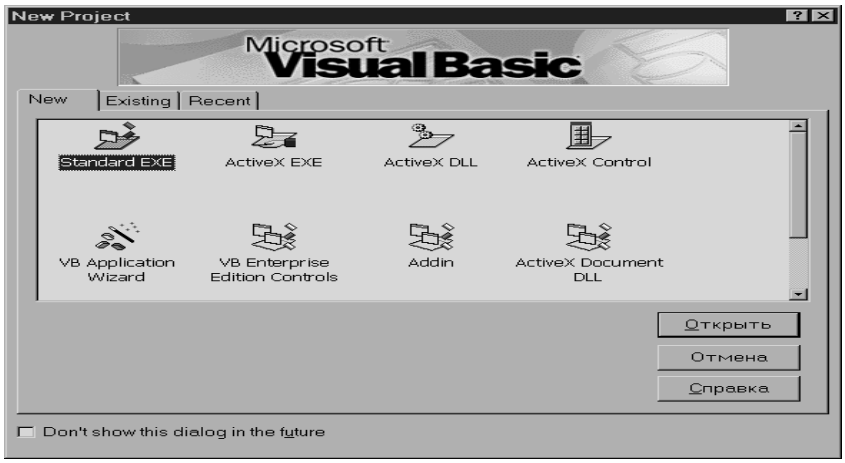


Рис. 11**.** Вікно нового проекту

У цьому вікні можна вибрати один з декількох типів шаблонів проектів, що полегшують початок роботи над додатком. Вікно складається із трьох вкладок**: New** (Нові проекти), **Existing** (Існуючі проекти) і **Recent** (Проекти, які недавно використовувалися).

Вибираючи **New**, ви доручаєте **VB6** створити основу додатка. Це заощаджує чимало часу. Згодом ви навчитеся користуватися багатьма з поданих тут шаблонів.

На вкладці **Existing** можна вибрати існуючий проект. Це може бути проект приклада, що входить у комплект **VB6,** або ж проект, над яким ви працювали в минулому. Чим більше ви будете працювати з **VB6**, тим частіше вам доведеться звертатися до цієї вкладки.

Вкладка **Recent** дозволяє вибрати з проектів, які недавно використовувалися, потрібний. Зовні вона схожа на Existing, але в ній перераховані лише ті проекти, над якими ви працювали останнім часом, а не всі існуючі.

Якщо ви бажаєте обходитися без вибору типу проекту – встановіть в нижній частині прапорець **Don't show this dialog in the future,** і при наступному запуску це вікно не з'явиться.

2.5.1. Настроювання середовища розробки VB6

Для настроювання середовища розробки програми VB використовується діалогове вікно **Options** (Параметри), що викликається з меню **Tools** (Сервіс) командою **Options** (Параметри). Вікно містить шість вкладок:

- **Editor** (Редактор);
- **Editor Format** (Формати редагування);
- **General** (Основні параметри налаштування);
- **Docking** (Інструменти середовища);
- **Environment** (Середовище проектування);
- **Advanced** (Розширені налаштування).

При зміні параметрів для збереження варіанта налаштування вийдіть з діалогового вікна **Options,** натиснувши кнопку **ОК**.

Для відмови від всіх здійснених на вкладках змін натисніть кнопку **Cancel (**Скасування**).**

Розглянемо більш докладно всі вкладки діалогового вікна **Options** і параметри, які настроюються з їхньою допомогою (рис. 12).

Вкладка *Editor*

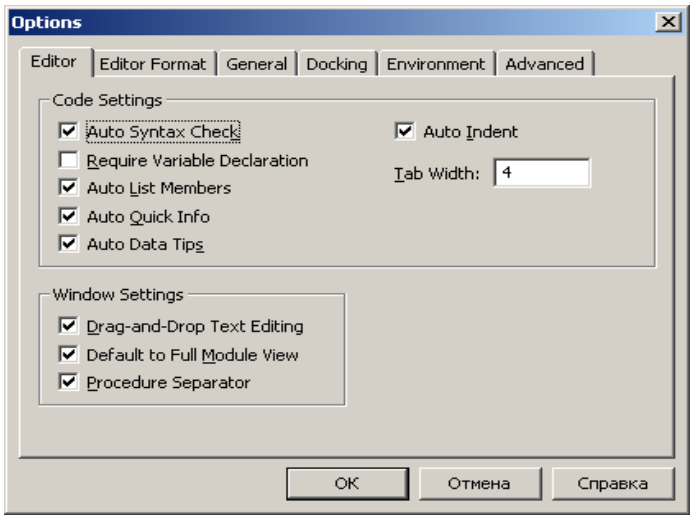


Рис. 12. Вкладка **Editor** (Редактор)

Найперша вкладка діалогового вікна **Options — Editor** (Редактор). Тут можна встановити параметри роботи редактора початкового коду додатка (рис. 12). Використовуючи прапорці групи **Code Settings** (Налаштування редактора коду), встановлюються такі параметри:

- **Auto Syntax Check** (Автоматичний контроль синтаксису) - автоматичний контроль синтаксису при введенні коду додатка в редакторі;
- **Require Variable Declaration** (Вимога оголошення змінних) - якщо цей прапорець установлений, будь-які змінні можна використовувати тільки після їхнього оголошення;
- **Auto List Members** (Автоматичне виведення списку можливих властивостей і методів об'єкта) - при введенні тексту програми методи і властивості відображуються в позиції курсора і їх можна легко вибрати;
- **Auto Quick Info** (Автоматичне виведення синтаксису функцій) — дозвіл або заборона виведення інформації про функції і їхні параметри при введенні тексту програми;
- **Auto Data Tips** (Автоматичне відображення значень) — під час відлагодження додатка відображається значення змінної, що знаходиться під покажчиком миші;
- **Auto Indent** (Автоматичне розміщення) — автоматичне розміщення відступу при введенні програми. Відступи роблять програму структурованою й зрозумілою при читанні (Для складних програм відступи просто обов'язкові).

У полі **Tab Width** вказується кількість позицій відступу при натисненні клавіші <TaЬ>. Початкове значення 8, але його можна змінити.

У групі **Window Settings** (Налаштування вікна) встановлюються параметри, пов'язані із фрагментами тексту:

- **Drag-and-Drop Text Editing** (Перетягування тексту) — дозволяє перетягування фрагментів тексту програми;
- **Default to Full Module View** (Відображення всіх процедур) — дозволяє відображати редакторові всі процедури виділеного об'єкта. Якщо цей прапорець не встановлений, процедури можна переглядати по черзі;

- **Procedure Separator** (Роздільник тексту процедур) — у режимі перегляду всіх процедур проекту встановлює між ними роздільник.

Вкладка *Editor Format* (Редактор формату)

На ній налаштовуються стилі тексту початкового коду додатка - шрифт, кольори, розмір тексту початкового коду залежно від призначення тексту коду. Використовуючи параметри даної вкладки, можна виділити кольорами, шрифтом і розміром шрифту основні елементи програми, наприклад, основний текст, текст коментарів, місця зупинки.

Для налаштування форматів елементів коду необхідно виконати такі дії:

- вибрати в області **Code Colors** зі списку типів програмного тексту тип, який потрібно використати;
- використовуючи списки, що розкриваються, **Foreground** і **Background**, установити кольори тексту й фону;
- за допомогою списків **Font** й **Size** задати найменування шрифту і його розмір;
- в області **Sample** переглянути отриманий результат здійснених змін;
- повторюючи пункти з 1-го по 4-й, змінити всі необхідні стилі тексту вікна редактора коду;
- для збереження встановлених параметрів натиснути кнопку ОК.

Вкладка *General*

На вкладці **General** діалогового вікна Options розміщені параметри, які призначені для керування загальними властивостями середовища проектування. В області **Form Grid Settings** (Налаштування сітки форм) знаходяться такі параметри для налаштування:

- **Show Grid** (Відображати сітку) — встановлює режим відображення ліній координатної сітки при розробленні макетів форм. Використовуючи розташовані під прапорцем поля **Width** (Ширина) і **Height** (Висота), можна встановити ширину й висоту комірок сітки;

- **Align Controls to Grid** (Прив'язати елементи керування до сітки) — задає прив'язку елементів керування до кроку сітки при розміщенні їх на макеті форми.

В області **Error Trapping** (Реакція на помилки) можна встановити один із трьох перемикачів, які задають варіант реакції додатка на помилки, які виникають під час його роботи:

- **Break on All Errors** — припиняти виконання додатка при виникненні будь-якої помилки;
- **Break in Class Module** — припиняти виконання додатка при виникненні будь-якої помилки в модулі класу;
- **Break on Unhandled Errors** — припиняти виконання додатка при виникненні будь-якої неопрацьованої помилки, тобто при такій помилці, для якої немає функції перехоплення.

В області **Compile** (Компіляція) можна встановити прапорці, що керують процесом компіляції програми:

- **Compile on Demand** (Компіляція в міру необхідності) — компіляція буде здійснюватися тільки для тих розділів програми, які необхідно виконувати. При цьому при виконанні команди **Start** (Запустити) з меню **Run** (Запуск) додаток буде запускатися швидше;
- **Background Compile** (Фонова компіляція) — цей прапорець дає можливість використовувати час очікування програми для її компіляції.

Крім перерахованих параметрів, на цій вкладці є ще два прапорці:

- **Show ToolTips** (Відображати підказки) — якщо цей прапорець встановлений, **VB6** відображає підказку з описом властивостей об'єкта на макеті форми, коли покажчик миші затримується на об'єкті на деякий час;

- **Collapse Proj, Hides Windows** (згорнути проект, сховати вікна) — цей прапорець змушує згортатись всі пов'язані із проектом вікна при згортанні вікна проекту.

Вкладка *Docking*

На вкладці **Docking** діалогового вікна **Options** всі прапорці об'єднані в одну групу **Dockable** . Ці прапорці визначають, які вікна середовища розробки можуть закріплюватися

("приклеюватися") до країв головного вікна програми VB6. Найменування вікон середовища розробки перераховані у вигляді прапорців. Налаштування полягає в установленні прапорців для відповідних їм вікон:

- **Immediate Window** — вікно Immediate (Безпосереднє виконання);
- **Locals Window** — вікно Locals (Локальні);
- **Watch Window** — вікно Watches (Спостереження);
- **Project Explorer** — провідник проекту;
- **Properties Window** — вікно властивостей;
- **Object Browser** — браузер об'єктів;
- **Form Layout** — вікно макета форм;
- **Toolbox** — панель елементів керування;
- **Color Palette** — палітра кольорів.

Вкладка *Environment*

Вкладка **Environment** діалогового вікна **Options** містить три області. В області **When Visual Basic starts** (Під час запуску **VB6**) можна встановити перемикачі:

- **Prompt for project** (Запрошення для вибору зразка проекту) — вказує на необхідність відкриття діалогового вікна **New Project** під час запуску VB6;
- **Create default project** — під час запуску VB6 створюється встановлений за попереднім визначенням тип проекту.

Область **When a program starts** (Під час запуску програми) керує можливістю збереження проекту під час запуску його на виконання.

Згодом при роботі з більшим проектом від збереження можна відмовитися для економії часу.

У даній групі перемикачів встановлюється одна з таких опцій:

- **Save Changes** — завжди зберігати зміни;
- **Prompt To Save Changes** — зберігати проект після попереднього запиту на збереження;
- **Don't Save Changes** — не зберігати здійснених змін.

Область **Show Templates For** (Показувати шаблони) містить список шаблонів, що з'являються у відповідних діалогових вікнах при створенні об'єктів. Використовуючи розташовані тут прапорці, можна відмовитися від частини шаблонів, які в цей момент не цікаві. Для цього необхідно зняти відповідний прапорець.

Наприклад**, якщо ви не бажаєте, щоб при створенні форм з'являлося діалогове вікно зі списком шаблонів форм, вам необхідно зняти прапорець **Form.

Поле **Template Directory** (Каталог шаблонів) вкладки призначає папку пошуку шаблонів за попереднім визначенням.

Вкладка *Advanced*

На вкладці **Advanced** діалогового вікна **Options** розташовані три прапорці, що мають таке призначення:

- **Background Project Load** — дозволяє завантаження проекту у фоновому режимі, що дає можливість під час завантаження іншого проекту продовжувати роботу над поточним;
- **Notify when changing shared project items** — повідомляє про зміну загального для декількох проектів об'єкта при одночасній роботі над цими проектами;
- **SDI Development Environment** — встановлює однодокументний інтерфейс (SDI, Single-Document Interface) середовища проектування. За попереднім визначенням використовується багатодокументний інтерфейс (MDI, Multiple-Document Interface).

Використовуючи поле **External HTML Editor** (Зовнішній редактор HTML) вкладки **Advanced,** можна встановити редактор для роботи з HTML-сторінками.

Інтегроване середовище розробки – **Integrated Development Environment (IDE)** (рис. 13) - важлива складова частина VB6. Саме тут ви збираєте воєдино додаток і проводите основний час у роботі над ним.

Вікно, на якому розташовується безліч піктограм різних елементів керування, називається **Toolbox** ("Палітра інструментів"). Вибравши натисканням миші на відповідній піктограмі потрібний елемент керування, ми встановлюємо його на формі проектованого додатка.

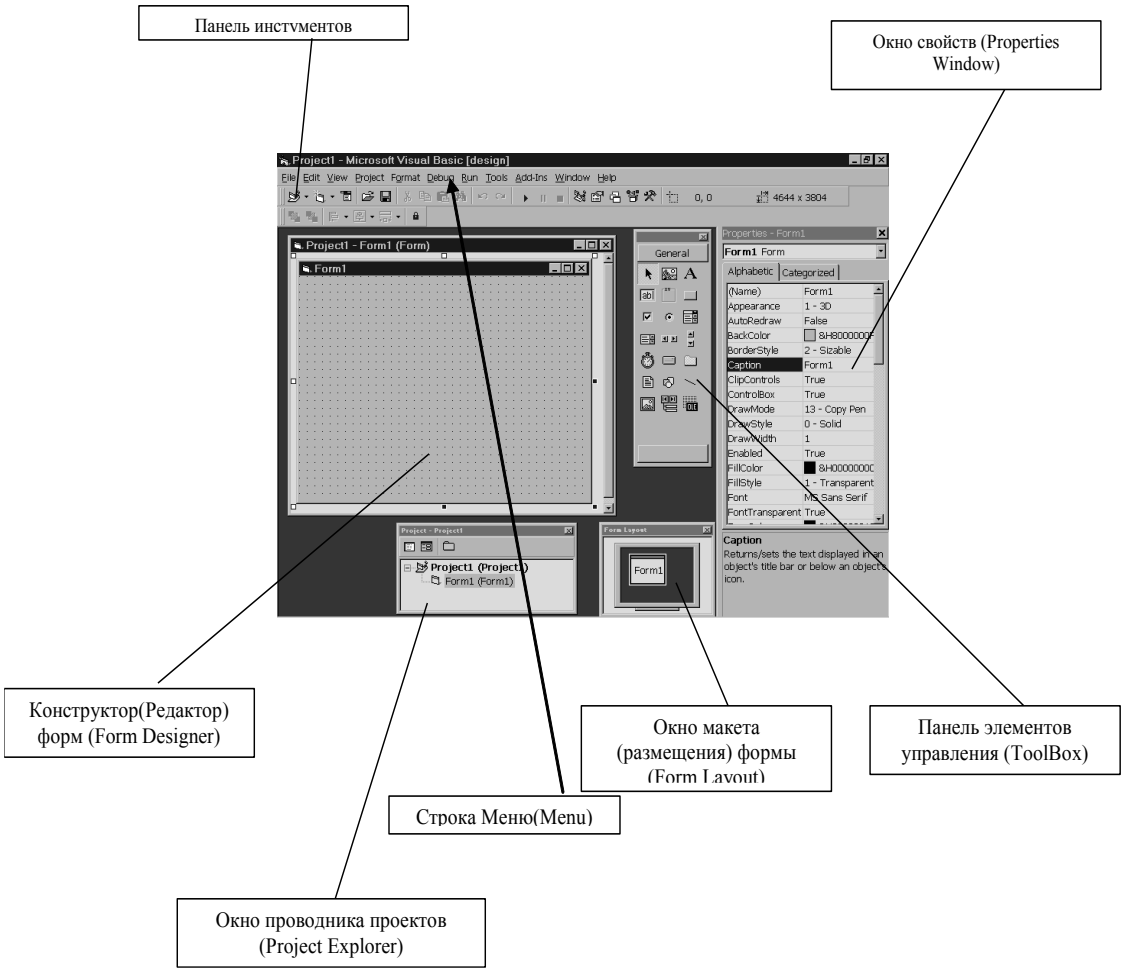


Рис. 13. Вікно розробки додатка

Встановлення полягає в тому, що ми розміщуємо покажчик миші в заздалегідь визначеній точці розташування на формі лівого верхнього кута елемента керування. Зафіксувавши ліву кнопку миші в натиснутому положенні, тягнемо прямокутний контур, що утвориться, до визначеної точки розташування правого нижнього кута елемента, де і відпускаємо.

Інший спосіб полягає у подвійному натисканні на піктограмі елемента в палітрі інструментів, після чого відповідний елемент з'являється в центрі активної форми, маючи деякі "стандартні" розміри. Далі його положення на формі та розміри регулюються за допомогою миші звичайним чином.

Основні доступні **елементи керування** наведені в табл. 8, **основні властивості** - в табл. 9-13. Значення властивостей об'єктів можуть бути встановлені розробником як на етапі розроблення додатка (за допомогою вікна властивостей), так і безпосередньо в програмному коді.

Таблиця 8

Елементи керування

Кнопка	Назва	Призначення
	**Pointer (Покажчик)**	**Використовується для позиціювання маркера (покажчика) миші**
	**PictureBox (Графічне вікно)**	**Розміщує на формі графічне вікно, призначене для об'єднання елементів у групи, для виведення в нього графічних зображень, а також тексту, графічних елементів і анімації**
	**Label (Мітка)**	**Розміщує на формі об'єкти, призначені для створення текстової інформації, написів і приміток**
	**TextBox (Текстове поле)**	**Розміщує на формі текстове поле, призначене для введення текстової інформації, чисел і дат**
	**Frame (Рамка)**	**Створює на формі рамку із заголовком для об'єднання об'єктів у логічну групу**
	**CommandButton (Кнопка керування)**	**Розміщує на формі кнопки керування для ініціації дій, виконання команд, запуску програм**
	**CheckBox (Прапорець)**	**Розміщує на формі прапорець, призначений для формування умов виконання програм або яких-небудь налаштувань, що працює за принципом "так - ні"**
	**OptionButton (Перемикач)**	**Створює у формі перемикачі для вибору режиму роботи або налаштування виконання програми**
	**ComboBox (Поле зі списком)**	**Створює на формі об'єкт, що містить одночасно поле введення й список, що розкривається**
	**ListBox (Список)**	**Створює на формі список для вибору одного або декількох значень із пропонованого списку значень**
	**HScrollBar (Горизонтальна смуга прокручування)**	**Розміщає у формі горизонтальну смугу прокручування, яка використовується як покажчик для вибору значення із заданого діапазону**
	**VScrollBar (Вертикальна смуга прокручування)**	**Розміщує на формі вертикальну смугу прокручування, яка використовується як покажчик для вибору значення із заданого діапазону**
	**Timer (Таймер)**	**Розміщує на формі таймер**
	**DriveListBox (Список пристроїв)**	**Створює на формі список пристроїв**
	**DirListBox (Список папок)**	**Створює на формі деревоподібний список папок**
	**FileListBox (Список файлів)**	**Створює на формі список файлів**
	**Shape (Обрис)**	**Створює у формі геометричні фігури, такі як прямокутник, квадрат, коло, еліпс, прямокутник і квадрат з округленими кутами**
	**Line (Лінія)**	**Створює лінії**
	**Image (Зображення)**	**Створює у формі поля, призначені для відображення графічних зображень**
	**Data (Дані)**	**Створює елемент керування даними в базі даних для переміщення по записах і відображення результату навігації**

Діалогове вікно **Properties** (Властивості) призначене для відображення і налаштування властивостей форми, а також розміщених на ній об'єктів. У ньому, наприклад, містяться такі властивості обраного об'єкта, як позиція на формі, висота, ширина, колір.

Вікно **Properties** викликається командою **Properties Window** (Вікно властивостей) з меню **View** (Вид), кнопкою **Properties Window** на стандартній панелі інструментів або командою **Properties** контекстного меню обраного об'єкта. Оскільки елементи керування кожний сам по собі є об'єктами, набір властивостей у цьому вікні змінюється залежно від обраного об'єкту. За допомогою вкладок **Alphabetic** (За абеткою) і **Categorized** (За категоріями) властивості об'єкта можна переглянути за абеткою або за групами (категоріями) відповідно.

У нижній частині вікна ви завжди знайдете підказку, що пояснює призначення обраного атрибута об'єкта.

Використовуючи діалогове вікно **Properties**, можна змінити попередньо встановлені властивості об'єктів. Частина властивостей об'єкта, наприклад, розміри й розташування, можна задати переміщенням і зміною його розмірів за допомогою миші в конструкторі форм. Як правило, форма містить багато об'єктів. Якщо вибрати одразу декілька об'єктів, то у вікні властивостей будуть відображені загальні для цих об'єктів властивості.

*По імені, зазначеному у властивості **Name** (Ім'я), об'єкт ідентифікується у формі і у тексті програми. Тому необхідно мати на увазі, що в одній формі не може бути двох об'єктів з однаковими іменами. Попередньо ця властивість встановлюється автоматично. Замість імені, заданого попередньо, краще використовувати ім'я, що відбиває його смислове значення.*

Розглянемо основні групи властивостей, подані на вкладці **Categorized** вікна **Properties.**

У групі **Appearance** (Оформлення) вікна **Properties** містяться властивості об'єкта, які задають атрибути його зовнішнього вигляду. Основні властивості даної групи наведені в табл. 9.

Властивості групи надають можливість редагувати об'єкт, встановлюють видимість, довжину даних в об'єкті (табл. 10).

Таблиця 9

Властивості об'єктів групи **Appearance**

Властивість	**Призначення**
Caption	**Задає текст у рядку заголовка об'єкта**
BorderStyle	**Задає стиль рамки об'єкта**
Palette	**Задає палітру кольору**
Picture	**Призначає значок, картинку для об'єкта. Використовуючи дану властивість форми, можна задати фонове графічне зображення**

Таблиця 10

Властивості об'єктів групи **Behavior**

Властивість	**Призначення**
Causes Validation	**Встановлює ознаку перевірки умови достовірності даних при виході з об'єкта**
Enabled	**Дозволяє або забороняє доступ до об'єкта**
MaxLength	**Встановлює максимальну довжину даних в об'єкті**
Visible	**Встановлює видимість об'єкта**

Група **Misc** призначена для задавання імені, тексту, що виводиться і т. ін. для об'єкту(табл. 11).

Таблиця 11

Властивості об'єктів групи **Misc**

Властивість	**Призначення**
Name	**Задає ім'я об'єкта**
Text	**Встановлює попередньо текст у поле**
Index	**Задає унікальний індекс об'єкта в колекції**

Група **Position** установлює положення і розміри об'єкта (табл. 12).

Таблиця 12

Властивості об'єктів групи **Position**

Властивість	**Призначення**
Left	**Задає положення об'єкта по горизонтальній осі від лівого краю форми або, у загальному випадку, від об'єкта-контейнера**
Top	**Задає положення об'єкта по вертикальній осі від його верхнього краю до верхньої сторони форми**
Width	**Задає горизонтальний розмір (ширину) об'єкта**
Height	**Задає вертикальний розмір (висоту) об'єкта**

Властивості групи **Scale** (Масштаб) встановлюють шкалу максимальних розмірів об'єктів у системі координат форми (табл. 13).

Таблиця 13

Властивості об'єктів групи **Scale**

Властивість	**Призначення**
ScaleLeft	**Задає максимальне положення об'єкта по горизонтальній осі**
ScaleTop	**Задає максимальне положення об'єкта по вертикальній осі**
ScaleWidth	**Задає максимальний горизонтальний розмір (максимальна ширина)**
ScaleHeight	**Задає максимальний вертикальний розмір (максимальна висота)**

2.5.2. Основні об'єкти VB6

2.5.2.1. Форма

Атрибутами об'єкта **ФОРМА** є біля 50 властивостей зазначеного об'єкта, які розташовуються у вікні властивостей (див. табл. 9-13).

Однією з найважливіших властивостей цього об'єкта є властивість **Name** (ім'я даного об'єкта). **ФОРМА** може мати ім'я, відмінне від імені, заданого VB6 попередньо.

Змістовні імена полегшують роботу з формами й іменами в програмі.

Розглянемо деякі методи, події й властивості, характерні для **ФОРМИ:**

- **Load** - оператор, що завантажує форму в пам'ять, але не відображає її на екрані;
- **Unload** - оператор, що вивантажує форму з пам'яті та видаляє її з екрана;
- **Show** - метод, що завантажує і показує форму на екрані;
- **Hide** - метод, який використовується для видалення форми з екрана, але не з пам'яті;
- **Activate** - подія, котра відбувається, якщо форма стає активною;
- **Deactivate** – подія, котра відбувається в тому випадку, коли форма перестає бути активною;
- **Resize** - подія, що відбувається при зміні розмірів форми.

Керуючі елементи або об'єкти керування

Об'єкти керування - це ті цеглинки, з яких створюється Windows-додаток.

2.5.2.2. Command Button

Command Button (командна кнопка) – одна з найпоширеніших об’єктів у Windows-додатках, використовуеться для вирішення будь-яких завдань: від найпростішого введення інформації до виведення спеціальних функцій, зв'язаних Windows-додатком.

Можна використовувати до 40 властивостей. Дві найважливіших з них:

- **Name;**
- **Caption.**

За значенням **Name** VB6 відрізняє одну кнопку від іншої. Якщо треба змінити напис на кнопці - використовується властивість **Caption**.

Керувати кнопкою можна за допомогою клавіші ***Alt+ підкреслена буква***. Крім того, користувач може перейти до кнопки клавішею **Tab**, а імітувати натискання лівої кнопки миші за допомогою клавіші “***Пропуск***”.

Важливою властивістю цього об'єкта є **Style**; попередньо заданим значенням цієї властивості є 0-**Standart**. Вибираючи значення властивості **Style**, користувач може залишити на кнопці тільки текст або помістити на кнопку якийсь рисунок (1-**Graphical**).

Розглянемо властивості, характерні тільки для **Command Button:**

- **Cancel** - ***True*** - для автоматичного виклику процедури **Click** при натисканні клавіші **Esc;**
- **Default** - ***True*** - для автоматичного виклику процедури **Click** при натисканні клавіші **Enter;**
- **DisabledPicture** - рисунок, зображуваний на кнопці, коли вона не доступна (властивість **Enabled**=***False***), якщо властивість **Style**=1;
- **DownPicture** - рисунок, зображений на кнопці, коли вона натиснута, якщо властивість **Style**=1;
- **Tabindex** - порядковий номер у послідовності переходу (при натисканні **Tab**);
- **ToolTipTex**t - спливаюча підказка для кнопки.

При запуску додатка, як правило, один з наявних на формі об'єктів повинний бути активним, тобто обробляти певним чином інформацію, одержувану від миші або клавіатури. У цьому випадку говорять, *що об'єкт має фокус*.

Наприклад, якщо є дві керуючі кнопки і одна з них має фокус, то натискання клавіші **Enter** призведе до виклику для неї процедури обробки події **Click.**

Якщо керуюча кнопка має фокус, то вона відображається з виділеною рамкою на формі.

У тому випадку, коли об'єкт одержує фокус, для нього виконується подія **GotFocus**. З іншого боку, при втраті фокуса об'єктом відбувається подія **LostFocus**.

Одержання фокуса об'єктом може бути реалізовано декількома способами:

- при натисканні на ньому лівою кнопкою миші;
- використовуючи клавіші переходу **Tab** або стрілки керування курсором;
- виконуючи для заданого об'єкта метод **SetFocus**

2.5.2.3. Label

Нагадаємо, що об'єкт **Label** містить текстову інформацію.

У випадку, коли текст, що знаходиться в мітці, не вміщається в одному рядку, то, за попереднім визначенням, він переноситься на такий рядок, якщо дозволяє задана висота елемента. В іншому випадку, та частина, що вийде за наявні межі, відображена не буде. Для автоматичного розширення мітки на формі необхідно встановити властивість **AutoSize**=***True***.

Якщо необхідно розташувати текст у мітці на декількох рядках з автоматичним збільшенням її висоти, варто дозволити перенос слів шляхом установки **WordWrap=*True***. Перенос, установлений таким чином, буде працювати, якщо встановлена властивість **AutoSize=*True***.

2.5.2.4. TextBox

Використовується для реалізації діалогу з користувачем шляхом введення ним із клавіатури певної інформації.

Зважаючи на те, що в елемента **TextBox** відсутня властивість **Caption**, для її заміни використовують мітку з відповідним текстом.

Текст, що вводиться користувачем, обробляється за допомогою властивості **Text**. Якщо він не вміщається в заданих межах текстового поля, можна дозволити перенос слів шляхом установлення властивості **MultiLine=*True***. Разом з тим необхідно вибрати один з варіантів для смуг прокручування тексту (властивість **ScrollBars**), тому що розмірів поля може не вистачити для відображення всієї введеної інформації.

Для того, щоб при одержанні фокуса текстовим полем курсор знаходився у заданій позиції (за попереднім визначенням - на початку тексту), використовується властивість

SelStart=<*позиція*>,

де <*позиція*> - порядковий номер символу в текстовому полі, перед яким буде розташована точка введення; при цьому нумерація символів починається з нуля. Для точки введення можна задати не тільки позицію курсора, але й кількість символів, які будуть виділені, тобто яку частину тексту необхідно замінити першим введеним символом. У цьому випадку засТОПовується властивість

SelLength=<*кількість*>.

Властивість **SelText =<*текст*>** використовується тоді, коли необхідно автоматично заміняти певну частину тексту. Коли поле отримає фокус, зазначена частина тексту в ньому зміниться на <***текст***>.

Слід зазначити, що властивості **SelStart, SelLength**, **SelText** доступні тільки в програмному коді і не можуть бути задані на етапі розроблення додатка.

2.5.2.5. Перемикачі (OptionButton)

Перемикачі використовуються в тому випадку, якщо необхідно вибрати одну з взаємовиключних можливостей.

Звичайно перемикачі групуються в рамках, але якщо використовується тільки одна група перемикачів, то їх можна групувати на формі.

Value – це властивість, яка використовується в режимі конструювання та у режимі виконання. Ця властивість може набувати два значення: **False** й **True**. У режимі конструювання властивості **Value** одній із кнопок можна присвоїти значення **True** (за попереднім визначенням **False**). Тоді цей перемикач буде встановлений при відкритті форми. За попереднім визначенням при відкритті форми буде встановлений тільки один перемикач. Перемикач має вигляд (див. табл. 8), але може набути й вигляду кнопки, якщо необхідно на даному перемикачі встановити яке-небудь зображення.

2.5.2.6. Списки (ListBox)

Список дозволяє користувачеві обирати один або декілька елементів з перерахованих. У будь-який час у список можна додавати нові елементи або вилучати існуючі. Якщо не всі елементи можуть одночасно відображатися в полі списку, то в ньому автоматично відображається смуга прокручування.

Основні властивості:

- **Name List –** здійснює формування списку. Назви пунктів списку вводяться в поле, що розміщене праворуч. Перехід на інший рядок здійснюється за допомогою комбінації клавіш **Ctrl+Enter**, якщо натиснути **Enter**, то вікно закриється. Редагування слів не припустимо.
- **Sorted –** за попереднім визначенням False, якщо встановити True, то автоматично відбудеться сортування списків за абеткою по зростанню не враховуючи регістр символів.
- **Columns** - дозволяє керувати кількістю стовпців у списку. Дана властивість може набувати значення: 0–однорядковий список; 1–багаторядковий список з горизонтальним прокручуванням.
- **Style**, за попереднім визначенням 0**– Standart**, 1**–Checkbox**– з'явиться прапорець.

Найчастіше використовують методи:

- **Add Item** - для включення рядків у список.
- **Clear** - для очищення вікна повністю (видаляє всі рядки).
- **Remove Item** - для видалення рядків зі списків.

2.5.2.7. Комбіновані поля (ComboBox)

Сполучають можливості текстового поля й списку. Використання списків пов'язане з однією потенційною проблемою, а саме вибір користувача обмежується рядками, що знаходяться в списках. Крім того, не можна прямо відредагувати рядок списку. Звичайні списки доцільні, якщо необхідно мати обмежений список і досить місця на формі .

ComboBox дозволяє вибрати зі списку певний рядок або ввести значення, якого немає в списку. Крім того, **ComboBox** може бути списком, що розкривається (займає менше місця на формі).

Існують три різновиди **ComboBox**, які вибираються в режимі конструювання:

- комбіноване поле, що розкривається;
- просте комбіноване поле;
- списки, що розкриваються.

Style:

0 – комбіноване поле, що розкривається, схоже на стандартне текстове поле, зі стрілкою праворуч, при цьому користувач може вибирати рядок зі списку, а може ввести свій текст. Цей варіант звичайно називається комбіноване поле;

1 - просте комбіноване поле являє собою різновид описаного; відмінність полягає в тому, що даний список постійно залишається відкритим;

2 – список, що розкривається, набуває рис **ListBox**, зовні схожий на комбіноване поле, але користувач обмежений тільки рядками, що входять у список. Редагування не припустиме.

2.5.2.8. Смуги прокручування (VscrollBar, HscrollBar)

Властивості смуг прокручування:

- **max** – визначає максимальне значення, яке набуває властивість **Value;**
- **min** - визначає мінімальне значення, яке набуває властивість **Value;**
- **Large Change** показує, як повинно змінитися значення властивості **Value,** коли користувач натискає на смузі прокручування;
- **Small Change** визначає зміну властивості **Value** при натисканні на одній зі стрілок смуги прокручування.

2.5.2.9. Вбудовані діалогові вікна

У проектах VB6 часто зустрічаються два різновиди діалогових вікон:

- вікна повідомлень;
- вікна введення.

Вікно повідомлення **MsgBox** виводить повідомлення для користувача.

Вікно введення **InputBox** дозволяє користувачеві вводити інформацію.

Функція **InputBox** відображає вікно діалогу, що дозволяє користувачеві вводити в текстовому полі дані, опис яких може бути заданий як значення одного з параметрів. Стандартними елементами такого діалогу є також керуючі кнопки: **ОК** (підтвердження дії) і **Cancel** (скасування дії).

Синтаксис **InputBox**:

InputBox (*символьний вираз1[,символьний вираз2[,символьний вираз3 [,число1[,число2]]*)

- ***символьний вираз1*** - текст (повідомлення), який відображається в діалоговому вікні;
- ***символьний вираз2*** - текст, який відображається в заголовку діалогового вікна;
- ***символьний вираз3*** - текст, який відображається в полі введення як повідомлення за попереднім визначенням;
- ***число1*** - відстань по горизонталі у твіпах між лівою межею вікна додатка й лівим краєм екрана;
- ***число2*** - відстань по вертикалі у твіпах між верхнєю межею вікна додатка й верхнім краєм екрана.

Якщо *число1* опущене, то вікно діалогу вирівнюється щодо центра екрана. Якщо *число2* опущене, то вікно діалогу розміщується по вертикалі приблизно на 1/3 висоти екрана.

Параметри, у квадратних дужках, можна не використовувати. Наявність ком, що відповідають відсутнім параметрам, обов'язкова.

Твіп - одиниця виміру відстані на екрані. Використовується для точної вказівки положення екранних елементів.
1 твіп= 1/20 пункту; 1 сантиметр = 567 твіпів.

Приклад

```
sd=InputBox("Введіть текст", "Вікно введення", "Ваше ім'я?",,5000)
```

Виконання програми припиняється до введення повідомлення (текстової або числової константи) і натискання кнопки **Ok**. Введене користувачем значення присвоюється змінній **sd**. Наведений ескіз вікна показує розміщення інформації в полі вікна діалогу й вікна в полі екрана. Вікно розташоване внизу з лівого боку екрана (рис. 14).

Рис. 14

Діалогове вікно виведення **MsgBox**

Синтаксис *MsgBox*:

MsgBox *символьний вираз1[,число[,символьний вираз2]*

символьний вираз1 - текст, який відображається як повідомлення в діалоговому вікні;

число - вид додаткових елементів, що вводять автоматично у вікно діалогу (табл. 14);

символьний вираз2 - текст, який відображається в заголовку діалогового вікна.

Таблиця 14

Число	Назва кнопок, значків
0	Ok
1	Ok, скасування
2	СТОП, повтор, пропустити
3	Так, ні, скасування
4	Так, ні
32	Значок ? (питання)
48	Значок ! (знак оклику)

Приклад. **MsgBox"Увага",4,"Вікно повідомлення" (рис. 15)**

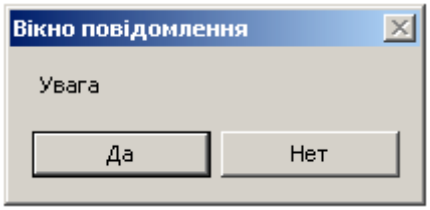


Рис. 15

Приклад. **MsgBox "Значення величини дорівнює"&t**

У полі вікна буде виведене значення змінної **t**, визначеної в результаті виконання діалогу за командою введення.

3. ЛАБОРАТОРНО-ПРАКТИЧНІ РОБОТИ

Схема курсу відповідно до положення про впровадження кредитно-модульної системи організації навчального процесу в УкрДАЗТ

Розділ “Основи програмування мовами високого рівня” є складовою частиною дисципліни "Обчислювальна техніка, програмування, моделювання систем", яка апробована на студентах першого та другого років навчання механічного факультету Української державної академії залізничного транспорту та охоплює матеріал 3,4 та 5-го тематичних модулів.

Методичні рекомендації з використання навчального посібника:

виконання лабораторно-практичних завдань варто починати після вивчення теоретичної частини відповідного розділу.

Всі пункти пропонованих завдань необхідно ретельно виконувати й зберігати створені документи й програми, тому що вони використаються в таких роботах.

На розсуд викладача найбільш підготовлені студенти можуть бути звільнені від виконання окремих найбільш простих завдань, а найменш підготовлені студенти можуть бути звільнені від виконання окремих найбільш складних завдань.

Тестові питання можуть бути використані студентами для самопідготовки, а викладачами для тематичного або модульного контролю знань студентів.

Вивчення теоретичного матеріалу

Навчальний посібник містить весь матеріал, необхідний студенту для засвоєння даних розділів курсу. На початку відповідного модуля студент отримує весь матеріал та самостійно вивчає теоретичні засади, розкриті в ньому (див.розд. 1,2).

Передбачається, що студент, ознайомлюючись з навчальним матеріалом, самостійно готується до лекційних занять, вивчаючи (допоміжну) навчально-методичну літературу та формулюючи список питань для обговорення з викладачем на лекції. У кінці навчального посібника наводиться перелік навчально-методичної літератури, яка буде корисною для підготовки до вирішення задач.

Викладачі можуть проводити заняття із застосуванням комп'ютерних технічних засобів навчання, використовуючи теоретичний матеріал 1-го та 2-го розділів навчального посібника.

Виконання лабораторних робіт

У розд. 3 наведено перелік питань та завдання, що виконуються студентами при виконанні лабораторних робіт.

Кожна робота відповідає певній темі лекційного матеріалу.

Студент на основі матеріалів виконаної лабораторної роботи оформлює звіт.

У процесі виконання лабораторної роботи студент здійснює такі види діяльності:

- відповідає на теоретичні (контрольні) запитання;
- виконує практичне (обов’язкове) завдання відповідно до варіанта. У кожній лабораторній роботі наводиться список обов'язкових для вирішення задач з поточної теми та варіанти індивідуальних завдань, які визначає викладач.

Звіт надається викладачеві безпосередньо після виконання лабораторної роботи.

Після виконання лабораторної роботи студентові виставляється оцінка. Складові оцінки за виконання роботи розподілені таким чином: 0-30 балів – відповіді на теоретичні питання та виконання обов’язкового завдання
0-60 балів – виконання індивідуального завдання, 0-10 балів – варіативність, оптимізація розв'язання задачі, висновки з роботи.

Виконання кожної лабораторної роботи має свою питому вагу в загальній оцінці. У табл. 15 наведені можливі вагові важелі для кожної лабораторної роботи.

Таблиця 15

Номер лабораторної роботи	Ваговий важіль	Номер лабораторної роботи	Ваговий важіль
1	**0,07**	**11**	**0,06**
2	**0,05**	**12**	**0,03**
3	**0,10**	**13**	**0,06**
4	**0,05**	**14**	**0,06**
5	**0,03**	**15**	**0,07**
6	**0,03**	**16**	**0,06**
7	**0,06**	**17**	**0,04**
8	**0,06**	**18**	**0,05**
9	**0,01**	**19**	**0,03**
10	**0,08**		

При недостатній сумарній кількості балів для одержання тієї підсумкової оцінки, що задовольняє студента, він має право набрати додаткові бали за рахунок самостійної роботи.

Виконання індивідуальних завдань (самостійна робота студента)

До кожної лабораторної роботи додаються набори завдань приблизно однакового рівня складності (див. розд. 4).

Використання тестових питань

Тести призначені як для контролю за навчальною діяльністю студента, так і для самопідготовки студента до виконання лабораторних робіт згідно з тематичними модулями.

Модуль 3. **Алгоритмічні мови програмування.** *Змістовий модуль 6.* Класифікація. Стислі характеристики алгоритмічних мов для сучасних ПЕОМ. Технологія роботи в середовищі програмування QBasic. *Змістовий модуль 7.* Дані алгоритмічної мови QBasic. *Змістовий модуль 8.* Основні оператори алгоритмічної мови QBasic.

Модуль 4. **Програмування однорідних та різнорідних структурованих даних.** *Змістовий модуль 9.* Обробка масивів даних. *Змістовий модуль 10*. Побудова інтерфейсу користувача.

Модуль 5. **Додаткові можливості QBasic.** *Змістовий модуль 11*. Модульне програмування. *Змістовий модуль* 12. Програмна обробка текстової інформаці *Змістовий модуль 13*. Файли на магнітних носіях. *Змістовий модуль 14*. Графічні можливості QBasic *Змістовий модуль 15*. Visual Basic.

Наприкінці тематичних блоків, після виконання лабораторних робіт, студент проходить тестування.

Перевірка рівня підготовки студента з даного тематичного блоку може відбуватися:

- за допомогою тестової програми. При апробації для тестування була обрана програма HyperTest 1.1. Програма достатньо проста у використанні, працює в мережі, видає результати для подальшої обробки, безкоштовно розповсюджується. Має три системи оцінювання, з яких була обрана така, що окремо враховує як вірні, так і невірні відповіді студента;
- із використанням роздаткового матеріалу.

Пакет завдань на електронному носії, розроблений викладачами кафедри обчислювальної техніки та систем управління, можна отримати на сайті кафедри: http://www.main.kart/kartsite/

Тестування з кожного тематичного блоку має свою питому вагу в загальній оцінці. Нижче наведені можливі вагові важелі для кожного блоку тестування.

Тестування з блоку змістових модулів 6-8 0,2
Тестування з блоку змістових модулів 9-10 0,1
Тестування з блоку змістових модулів 11-14 0,4
Тестування з блоку змістових модулів 15 0,3

Загальна оцінка

При викладанні дисципліни "Обчислювальна техніка, програмування, моделювання систем" у відповідних модулях була запропонована така система оцінювання знань студента: за модуль студент отримує загальну оцінку, яка складається із середньозваженої оцінок за такі види діяльності, враховуючи вагові важелі:

Вид діяльності	**Вагові важелі**
Виконання лабораторної роботи	0,85
Тестування	0,15

РОБОТА 1
ОЗНАЙОМЛЕННЯ ІЗ СЕРЕДОВИЩЕМ ПРОГРАМУВАННЯ QBASIC

Контрольні питання

1. Як завантажити для роботи середовище QBasic?
2. Ознайомтесь зі складовими частинами вікна середовища QBasic.
3. Вкажіть призначення кожного з пунктів меню QBasic:

Файл, Редактирование, Просмотр, Запуск, Отладка, Параметры, Справка:

- Вкажіть призначення команди *Новый* з меню *Файл*?
- Вкажіть призначення команди *Открыть* з меню *Файл*?
- Вкажіть різницю між командами *Сохранить* та *Сохранить как*....з меню *Файл*?
- Які вимоги висуває середовище QBasic (MS DOS) до імені файла?

4. Яке розширення (за попереднім визначенням) текстам програм надає QBasic?
5. Вкажіть призначення команд *Вырезать, Копировать, Вставить* з меню *Редактирование*?

6. Опишіть технологію копіювання фрагмента тексту програми.

7. Запишіть різницю між командами *Запуск, Перезапустить* та *Продолжить* з меню *Запуск*?

8. Які розділи містить довідкова система?

9. Як вийти з довідкової системи?

10. Натискуванням яких клавіш можна перервати виконання Basic-програми?

11. Опишіть призначення клавіш або їх сполучення:

Shift+DEL, Ctrl+Insert, Ctrl+Y, Shift+Insert, DEL, Backspace, Shift+F5 , F5, Esc.

12. Для чого призначене вікно безпосереднього виконання?

13. Комбінацією яких клавіш відбувається перемикання розкладки клавіатури на вашому ПК між кирилицею та латинню?

14. Наберіть на клавіатурі ПК такий текст:

```
REM Робота з текстовим редактором у середовищі QBasic
REM Це рядок коментарю
' Це теж рядок коментарю
REM Такий оператор - введення значення змінної X
INPUT "Ввести значення X="; X
REM Такий оператор присвоювання
Z=A*X^2+B*COS(2*X)-LOG(X)
REM Такий оператор виведення інформації
PRINT "РЕЗУЛЬТАТИ ОБЧИСЛЕНЬ X=";X,"Y=";Y
END 'Кінець програми
```

- Виділіть весь текст, скопіюйте його до буферної пам'яті та вставте його нижче набраного тексту.
- Збережіть набраний текст у вигляді програми на диску.
- Перевірте, чи ваша програма збереглася на диску.

15. У режимі безпосереднього виконання обчисліть і запишіть результат для такого арифметичного виразу:

3.478^5 + 7.921*24.718 – 6.2753E-2 + SQR(2256.44) - SIN(2.84)

16. У режимі довідки (Предметный указатель) виберіть послідовно оператори CIRCLE та PLAY (оператор). Скопіюйте наведені в описах цих операторів приклади в робоче вікно та запустіть їх на виконання.

17. Коректно завершіть роботу в середовищі QBasic.

РОБОТА 2

ПРОГРАМУВАННЯ АЛГОРИТМІВ ЛІНІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Контрольні питання

1. З яких груп символів складається алфавіт мови Basic?
2. У чому полягає різниця між константою і змінною?
3. На які типи поділяються константи мови Basic? Вкажіть діапазони їх значень.
4. Визначте поняття «ідентифікатор». Наведіть правила складання ідентифікаторів.
5. Яких типів можуть бути змінні? Як можна задати в програмі тип змінної? Наведіть приклади.
6. На які групи розподіляють оператори за призначенням?
7. Що таке вбудовані (стандартні) функції? На які групи вони розподілені? Наведіть приклади запису найбільш поширених функцій.
8. Вкажіть пріоритет виконання операцій при обчисленні арифметичного виразу?
9. Призначення та формат запису оператора присвоєння.
10. Формат запису оператора виведення. Зонне та компактне уявлення даних, що виводяться.
11. Формат запису та призначення операторів **TAB, SPC, LOCATE.**
12. Наведіть програмні фрагменти, що задають вказані далі значення змінним з довільними ідентифікаторами за допомогою: операторів **INPUT,** операторів присвоювання, операторів **DATA, READ**

3.4, -0.666, $25*10^8$, 0, колесо

13. Запишіть Basic - програму, вихідні дані та результати розв'язання задачі згідно з варіантом індивідуального завдання.

РОБОТА № 3

ПРОГРАМУВАННЯ РОЗГАЛУЖЕНИХ АЛГОРИТМІВ З ВИКОРИСТАННЯМ ОПЕРАТОРІВ ПЕРЕДАЧІ УПРАВЛІННЯ

Контрольні питання

1. Перерахуйте логічні функції, які використовуються в логічних виразах операторів управління. Що є результатом їх виконання?
2. Які умовні оператори ви знаєте?

3. Які дві форми запису умовного оператора **IF** ви знаєте? Чим вони відрізняються?

4. Який оператор називається оператором множинного вибору? Як він працює?

5. Наведіть формат запису оператора **SELECT CASE**. Що таке вираз вибору в цьому операторі?

6. Перерахуйте особливості використання оператора безумовного переходу **GOTO**?

7. Наведіть 3 можливі варіанти фрагмента програми організації розгалуження (обчислення значення **y**) з використанням лінійної форми оператора **IF**:

$$y = \begin{cases} 1; & якщо \quad x < 5 \\ 0; & якщо \quad x \geq 5 \end{cases}$$

8. Запишіть логічний вираз, що відповідає умові:

15<=x<27 і a>3+d або d=100
z>5 або a+z<-3 і 5<x<y та v≠30
¬y>x і a^2+5<d-3 або sin(y)=e^x

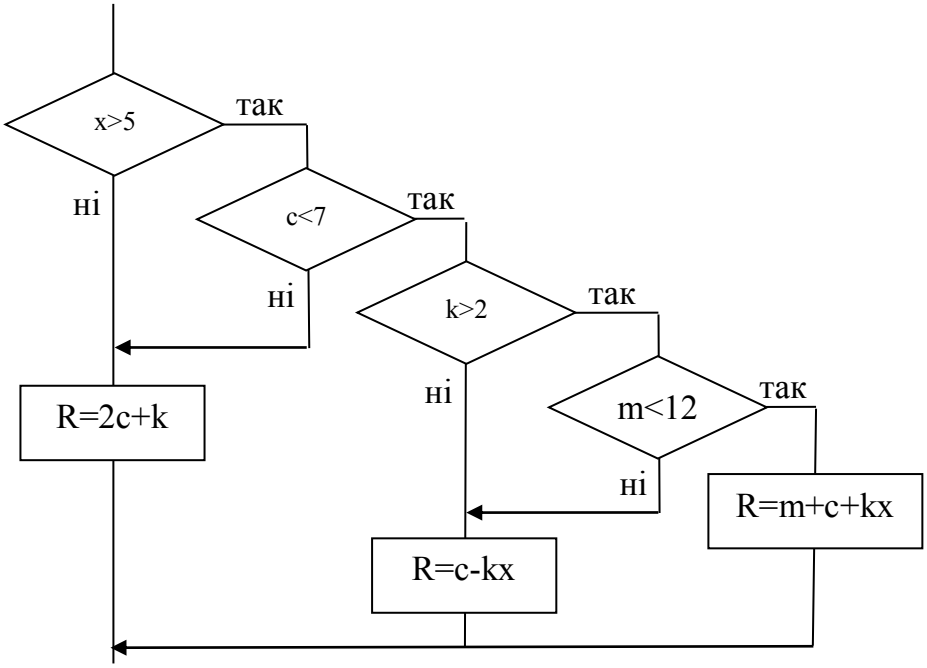
9. Наведіть фрагменти програми організації розгалуження (обчислення значення **y**) з використанням:

- блочної форми оператора **IF;**
- оператора **SELECT CASE**

$$y = \begin{cases} \ln(x^5 + x + 1) & якщо \quad 1 < x \leq 3 \\ x^5 + x + 1; & якщо \quad 3 < x < 5 \\ \cos(x^5 + x + 1); & якщо \quad 5 \leq x < 7 \\ \dfrac{1}{x^5 + x + 1;} & якщо \quad x \geq 7 \ або \ x \leq 1 \end{cases}$$

10. Складіть Basic – програму з використанням оператора **ON….GOTO**, яка передбачає введення чисел від 1 до 3 та виведення повідомлення, що введено відповідне число. Якщо введене число поза діапазоном, передбачте відповідне повідомлення та повторне введення. Програма повинна закінчувати роботу, якщо введене число 99.

11. Запишіть фрагмент програми для наведеної далі схеми алгоритму



12. Запишіть Basic - програми, вихідні дані та результати розв'язання задач згідно із завданням.

РОБОТА № 4

ПРОГРАМУВАННЯ АЛГОРИТМІВ ПРОСТИХ АРИФМЕТИЧНИХ ЦИКЛІЧНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Контрольні питання

1. Які оператори, що реалізують циклічні алгоритми, ви знаєте?

2. Наведіть формати оператора **DO... LOOP**, що реалізують базові структури **ЦИКЛ - ПОКИ, ЦИКЛ - ДО**. Чим відрізняється цикл із передумовою від циклу із постумовою?

3. Назвіть дії, що реалізуються в процесі виконання оператора **FOR ... NEXT.**

4. Яких правил необхідно дотримуватися при використанні оператора **FOR ...NEXT?**

5. Запишіть 2 варіанти програми, що реалізує обчислення факторіала числа n (n>0) A = n!, використовуючи конструкції:

1) DO... LOOP; 2) **FOR ... NEXT.**

6. Наведіть програму, що обчислює значення функції y=cos(x)+2, де x∈[0;1.7] hx=0.1/

7. Наведіть програму, що обчислює та виводить значення функції **y**

$$y = \sum_{i=1}^{10} x^i + \prod_{k=a}^{b} (k+1)$$

$$t1 = \sum_{n=1}^{5} n^2 \quad t2 = \prod_{j=1}^{5} (j+7) \quad t3 = (t1 + t2)!$$

8. Наведіть Basic-програму, що обчислює та виводить на екран монітора значення: a=Ln(b+2), де $b \in [t1;t2]$ $h_b=0.3$, за умови, що t1=2.5; t2=3.9

9. Запишіть Basic-програми, вихідні дані та результати розв'язання задач згідно з варіантом індивідуального завдання .

РОБОТА 5
ПРОГРАМУВАННЯ АЛГОРИТМІВ ВКЛАДЕНИХ АРИФМЕТИЧНИХ ЦИКЛІЧНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Контрольні питання

1. Наведіть приклад (фрагмент програми), в якому для реалізації вкладеного арифметичного циклічного процесу застосований оператор **FOR ... NEXT.**

2. Запишіть у відповідній послідовності результати виконання програми, які будуть на екрані монітора

```
For I=2 To 3
 For J=4 To 6
  Print I;“* “;J;“=“;I*j
  Next J
  Print
Next I
```

3. Як можна записати оператор **FOR ... NEXT** для реалізації вкладених циклів, якщо декілька циклів мають спільну кінцеву точку?

4**.** Чим відрізняється застосування оператора **DO... LOOP** від **FOR ... NEXT** для п.3.?

5. Запишіть 2 варіанти програми, що реалізує обчислення значення функції **y**

$$y = \sum_{x=5}^{8} x!$$

використовуючи оператори: **FOR ... NEXT, DO... LOOP.**

6. Запишіть Basic-програму, що реалізує обчислення за допомогою оператора **DO... LOOP** з післяумовою значення функції **y**

$$y = \sum_{x=5}^{8} 4 \prod_{i=3}^{5} x^i$$

7. Запишіть Basic-програму, що реалізує обчислення функції **y** за допомогою оператора **DO... LOOP** з передумовою

$$y = \sum_{x=5}^{8} \prod_{i=3}^{5} x^i$$

8. Наведіть Basic-програму, що обчислює значення функції **y = cos(x)+2*t,** якщо $x \in [0;1.7]$ $h_x=0.1$, $t \in [1.4]$ $h_t=0.7$, використовуючи оператори циклів, обрані за власним бажанням.

9. Запишіть Basic-програми, вихідні дані та результати роозв'язання задач згідно з варіантом індивідуального завдання.

РОБОТА 6
ПРОГРАМУВАННЯ АЛГОРИТМІВ ІТЕРАЦІЙНИХ ЦИКЛІЧНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Контрольні питання

1. Які оператори циклу недоцільно використовувати для реалізації управління в ітераційних алгоритмах?
2. Наведіть приклади програмної реалізації рекурентних співвідношень, що можуть входити до складу ітераційних циклів.
3. Наведіть Basic-програму обчислення значення функції **Y**, яка є сумою елементів нескінченого числового ряду, що збігається:

$$y = \frac{2}{1+a} - \frac{3}{2+a^2} + \frac{4}{3+a^3} - \frac{5}{4+a^4} + \cdots + \frac{i+1}{I+a^i} + \cdots$$

4. Наведіть фрагмент Basic-програми, що для задачі п.3, що підраховує кількість елементів суми, більших значення змінної **a** та менших **b**, де **a, b** - довільні числа.
5. Як в програмі п.3 здійснити виведення усіх елементів, що увійшли до суми? Наведіть потрібний оператор та вкажіть його місцезнаходження.
6. Запишіть Basic-програму, що реалізує обчислення квадратного кореня додатного числа **X.**
7. Наведіть Basic-програми, вихідні дані та результати розв'язання задач згідно з варіантом індивідуального завдання.

РОБОТА 7

ПРОГРАМУВАННЯ АЛГОРИТМІВ ВИЗНАЧЕННЯ НАЙБІЛЬШОГО ТА НАЙМЕНШОГО ЗНАЧЕННЯ ФУНКЦІЇ

Контрольні питання

1. Вкажіть, які оператори можна використовувати для організації управління в алгоритмі пошуку екстремуму функції на відрізку.

2. Наведіть умову задачі та фрагмент програми, коли для обчислення екстремуму функції, що залежить від двох змінних, використовуються вкладені цикли.

3. Наведіть умову задачі, в якій треба знайти максимальне та мінімальне значення функції, та складіть фрагмент програми для її розв'язання:

- використовуючи два цикли;
- використовуючи один цикл.

4. Наведіть програмну реалізацію алгоритму пошуку максимального (**Ymax)** значення функції Y = sin (x)-cos(x+1) (глобального max) на відрізку [a; b] та визначення значення аргументу **(Xmax)**, при якому досягається цей максимум.

5. Для задачі п.4 складіть алгоритм і Basic-програму пошуку порядкових номерів максимального та мінімального значень функції

6. Наведіть Basic-програми, вихідні дані та результати розв'язання задач згідно з варіантом індивідуального завдання.

РОБОТА 8

ПРОГРАМУВАННЯ АЛГОРИТМІВ ОБРОБКИ ОДНОВИМІРНИХ МАСИВІВ ДАНИХ

Контрольні питання

1. Визначте поняття:

- масив та ідентифікатор масиву в Basic-програмі;
- опис масиву в Basic-програмі.

2. Наведіть фрагмент Basic-програми:

- введення масиву **A,** що містить 7 елементів;
- виведення масиву **V**, що містить **n** елементів;
- обчислення суми S

$$S = \sum_{i=1}^{N} x_i$$

- обчислення добутку P

$$P = \prod_{j=1}^{M} y_j$$

3. Складіть Basic-програму формування масиву **Z** , що містить **k** дійсних чисел в діапазоні [a; b] за допомогою генератора випадкових чисел RND. Підрахуйте кількість елементів масиву **{ z_j },** які більші середнього арифметичного елементів цього масиву.

4. Складіть Basic-програму формування масиву **C** з елементів масиву **D,** який містить **m** елементів. Підрахуйте кількість парних елементів у масиві **C**

$$c_i = \begin{cases} d_i; & якщо \quad d_i \le 0 \\ \dfrac{1}{d_i}; & якщо \quad d_i > 0 \end{cases}$$

5. Складіть Basic-програму формування масиву **{ y_j }** з елементів **$2<x_i<15$,** масиву **{x_i},** i=l,k**,** та виведіть на екран монітору кількість елементів у масиві **{ y_j }.**

6. Складіть Basic-програму обчислення найменшого серед елементів масиву **{z_i},** i=1,10 з непарними індексами.

7. Наведіть Basic-програми, вихідні дані та результати розв'язання задач згідно з варіантом індивідуального завдання.

РОБОТА 9
ПРОГРАМУВАННЯ АЛГОРИТМІВ ОБРОБКИ ДВОВИМІРНИХ МАСИВІВ ДАНИХ

Контрольні питання

1. Визначте поняття:
   - двовимірний масив;
   - кількість елементів двовимірного масиву.
2. Наведіть фрагмент Basic – програми:
   - введення масиву **A,** що містить 5 рядків та 4 стовпці;
   - виведення масиву **B**, що містить **n** рядків та **m** стовпців;
   - обчислення суми S

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

- обчислення добутку P

$$P = \prod_{j=1}^{M} \prod_{i=1}^{N} Y_{ji}$$

3. Складіть Basic-програму , що обчислює:

- кількість ненульових елементів, розташованих в непарних стовпцях масиву **{f_{ij}},** i=1; n; j=1; m;
- різницю між найбільшим і найменшим значеннями елементів масиву **{b_{ij}},** i=1;4; j=1;3;
- кількість елементів парних рядків масиву **{c_{ij}},** i=l;n; j=1;k, які задовольняють умові **b < c_{ij} < d.** (b, d – довільні числа);
- середнє арифметичне елементів масиву **{$x_{i,j}$},** **i**=1;n;j=1;m (m=n), розташованих на головній та побічній діагоналях.

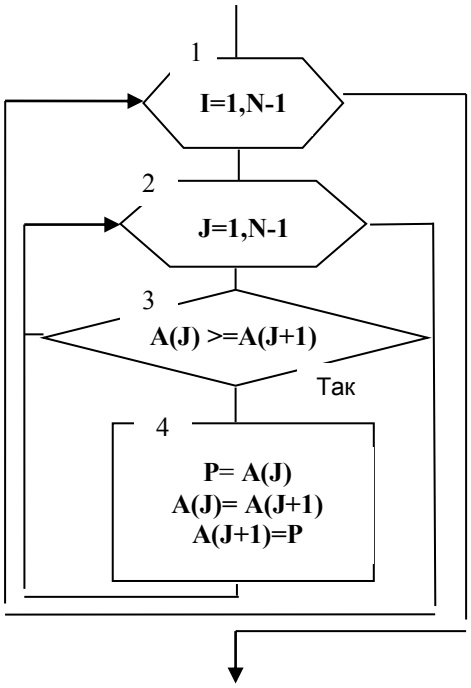
Наведіть Basic-програми, вихідні дані та результати розв'язання задач згідно з варіантом індивідуального завдання.

РОБОТА 10

ПРОГРАМУВАННЯ АЛГОРИТМІВ УПОРЯДКУВАННЯ ЕЛЕМЕНТІВ МАСИВІВ ЗА ЗАДАНОЮ КЛЮЧОВОЮ ОЗНАКОЮ

Контрольні питання

1. Від яких чинників залежить ефективність роботи алгоритму упорядкування масиву та відповідної програми?
2. Який зв’язок між поняттями “упорядкування” та “сортування” масиву?
3. Як здійснюється упорядкування за алфавітом символьних даних?
4. Складіть програму фрагмента алгоритму:
   - сортування “Вибором”;
   - сортування методом “Бульбашки”;
   - сортування поліпшеним методом “Бульбашки”.
5. Запишіть призначення блоків та програму фрагменту алгоритму “Наївне сортування”. Описати призначення блоків. Порівняти цей метод з іншими відомими вам



6. Наведіть Basic-програми, вихідні дані та результати розв'язання задач згідно з варіантоміндивідуального завдання.

РОБОТА 11
РОЗРОБЛЕННЯ BASIC – ПРОГРАМ З ВИКОРИСТАННЯМ ПРОЦЕДУР – ФУНКЦІЙ

Контрольні питання

1. Який підхід у програмуванні називають структурним?
2. Які правила використання об’яви процедур–функцій за допомогою оператора DECLARE?
3. За якими правилами записуються імена процедур–функцій, навести приклади запису.
4. Яким оператором закінчується запис процедури – функції?
5. Як позначається передача масиву в якості параметра в процедуру – функцію?
6. Для чого використовують оператори COMMON та SHARED в зовнішніх процедурах ?
7. Чим відрізняються між собою формальні та фактичні параметри при використанні процедур- функцій?
8. Коли не можна використати процедуру–функцію, а потрібна процедура-підпрограма ?

9. Де зберігаються процедури–функції в середовищі QBasic і як до них звертатися для редагування?

10. Наведіть головну Basic-програму, процедуру-функцію, вихідні дані та результати розв'язання задачі згідно з варіантом індивідуального завдання.

РОБОТА 12
РОЗРОБЛЕННЯ BASIC – ПРОГРАМ З ВИКОРИСТАННЯМ ПРОЦЕДУР- ПІДПРОГРАМ

Контрольні питання

1. Наведіть визначення процедури-підпрограми.
2. Які правила використання об’яви процедури-підпрограми за допомогою оператора DECLARE?
3. Вкажіть загальну форму опису процедури-підпрограми SUB.
4. Як відбувається звернення до процедури-підпрограми?
5. Як відбувається повернення результатів з процедури-підпрограми?
6. Яка відповідність повинна бути між формальними і фактичними параметрами в процедурах- підпрограмах?
7. Як можна організувати досимвольний вихід із зовнішньої процедури?
8. Наведіть головну Basic-програму, процедуру-підпрограму, вихідні дані та результати роз'язання задачі згідно з варіантом індивідуального завдання.

РОБОТА 13
РОЗРОБЛЕННЯ BASIC – ПРОГРАМ З ВИКОРИСТАННЯМ ТАБЛИЧНИХ ФОРМ

Контрольні питання

1. За допомогою якого оператора Basic виконується форматне виведення даних?
2. Запишіть формат оператора PRINT USING.
3. Що являє собою форматний рядок?
4. Що являє собою список виразів оператора PRINT USING?
5. Які символи формату використовуються в операторі PRINT USING? Їх призначення.

6. Наведіть:

- таблицю ідентифікаторів, що використовуються для схеми алгоритму та програми розв'язання задачі згідно з варіантом індивідуального завдання;
- схему алгоритму розв’язання індивідуальної задачі;
- Basic-програму, вихідні дані та результати розв'язання задачі згідно з варіантом індивідуального завдання.

РОБОТА 14
РОЗРОБЛЕННЯ BASIC – ПРОГРАМ ОБРОБКИ СИМВОЛЬНИХ ДАНИХ

Контрольні питання

1. Що таке символьна константа? Які способи задавання символьних констант у програмі ви знаєте?
2. Наведіть формат опису символьних змінних у програмі.
3. Поясніть особливості використання в описі символьних змінних параметра довжини.
4. Чому символьна змінна займає в пам'яті на 1 байт більше, ніж зазначено в описі довжини?
5. З яких складових будуються символьні вирази?
6. Які операції припустимі над строковими даними?
7. За якими правилами здійснюється порівняння рядків?
8. Як здійснюється доступ до окремого символу рядка?
9. Із заданого рядка одержати новий, замінивши всі коми на крапки. Вивести отриманий рядок.
10. Складіть програму, що друкує задане слово задом наперед.
11. Наведіть Basic-програму, вихідні дані та результати розв'язання задачі згідно з варіантом індивідуального завдання.

РОБОТА 15
РОЗРОБЛЕННЯ BASIC – ПРОГРАМ , У ЯКИХ ЗАСТОСОВУЮТЬСЯ ФАЙЛИ НА МАГНІТНИХ НОСІЯХ

Контрольні питання

1. Які переваги надає збереження файлів на магнітних носіях?
2. Дайте визначення фізичного та логічного записів. Чим вони відрізняються ?

3. Що таке “метод доступу” до інформації у файлі? Стисло охарактеризуйте існуючі методи доступу**:**

- послідовний;
- прямий (довільний).

4. Перерахуйте обов'язкові кроки, які виконуються в програмі при роботі з файлами.

5. Які оператори потрібні, щоб створити:

- послідовний файл;
- прямий файл.

6. Які оператори потрібні, щоб прочитати:

- послідовний файл;
- прямий файл.

7. Що визначає параметр “*режим”* при роботі з послідовним файлом? Які значення він набуває?

8. Наведіть формат оператора **FIELD**.

9. Порівняйте логіку роботи операторів **PRINT WRITE**.

10. Наведіть Basic-програму, вихідні дані та результати розв'язання задачі згідно з варіантом індивідуального завдання.

РОБОТА 16
ГРАФІЧНІ МОЖЛИВОСТІ QBASIC

Контрольні питання

1. Чим характеризуються графічні режими екрана?
2. Перерахуйте параметри оператора SCREEN. Вкажіть їх призначення.
3. Стисло охарактеризуйте поняття:
   - растрова графіка;
   - векторна графіка.
4. Що таке бітова глибина?
5. Які існують типи координат графічного екрана? Стисло охарактеризуйте їх.
6. Наведіть формат запису оператора **LINE**. Перерахуйте його параметри та вкажіть їх призначення.
7. Для чого призначена команда **DRAW** ?
8. Наведіть формат запису оператора **CIRCLE**. Перерахуйте його параметри та вкажіть їх призначення.
9. Наведіть алгоритм імітації руху зображення об'єкта на екрані.
10. Наведіть Basic-програму, вихідні дані та результати розв'язання задачі згідно з варіантом індивідуального завдання.

РОБОТА 17

ОЗНАЙОМЛЕННЯ ІЗ СЕРЕДОВИЩЕМ ПРОГРАМУВАННЯ VISUAL BASIC. ПРОГРАМУВАННЯ ЛІНІЙНИХ АЛГОРИТМІЧНИХ СТРУКТУР

Завдання 1. Створіть проект, в якому розрахуйте кількість локомотивів, що знаходяться в експлуатації, в резерві, в ремонті.

Вихідні дані. Загальна кількість локомотивів. У резерві – 10% від загального, в експлуатації –60%.

Результат. Вивести на формі кількість локомотивів, які експлуатуються, резерв і кількість локомотивів в ремонті.

Обов'язкові умови. Введення вихідних даних організувати за допомогою об'єкта TextBox , розрахунки і виведення даних повинні виконуватися при натисненні відповідної кнопки.

Порядок виконання роботи

1. Увійдіть в середовище програмування **Visual Basic 6.0**. Створіть новий проект (**Standard Exe**). Назвіть ваш проект «**Lab_1**». Для цього у вікні проекту (**Project**) виберіть перший рядок (рядок проекту) і відкрийте вікно властивостей командою **View =>Properties Window**. У вікні, яке з'явилося у властивості **Name** вказати **Lab_1.**

2. Створіть форму і розмістіть на ній об'єкти: **TextBox** , **Label** (для виведення написів над вікном введення даних і вікнами виведення результатів), **CommandButton** і **PictureBox**. Для розміщення об'єктів на формі скористайтеся таким методом:

- у вікні інструментів (**Toolbox**) виберіть необхідний об'єкт;
- зробіть подвійний клік на вибраному об'єкті;
- об'єкт, що з'явився на формі, розмістіть у вибраному вами місці на формі. Які ще методи розміщення об'єктів на формі ви знаєте?

3. У вікні властивостей відредагуйте властивості форми та об'єктів, розміщених на ній, таким чином:

- в **заголовку форми** - напис «**Лабораторна робота**» (властивість **Caption** об'єкта **Form1**), колір фону будь-який (властивість **BackColor** об'єкта **Form1**), виведення форми здійсніть в центрі екрана (скористайтеся вікном розміщення форм **Form Layout**);
- об'єкт **Label1 (Мітка)** - «Введіть кількість локомотивів в парку» (властивість **Caption**);
- об'єкт **Label2** - «Кількість локомотивів в резерві» ;
- об'єкт **Label3** - «Кількість локомотивів в експлуатації»;

- об'єкт **Label4** - «Кількість локомотивів в ремонті»;
- об'єкти **TextBox (Текстове поле)** мають бути втоплені (властивість **Border Style - 1**) і не містити напису(властивість **Text**), колір фону – жовтий (властивість **BackColo**r);
- **Command1(Командна кнопка)** – «Розрахунок»;
- **Command2** – «Вихід»;
- **PictureBox** – рисунок з каталогу «Малюнки».

4. Запустіть додаток на виконання (режим **Run-time**). Для цього необхідно натиснути клавішу F5.

5. Методом поперемінного перемикання в режим Run-time і назад (режим конструктора) дослідіть і опишіть такі властивості об'єктів як: розмір, місцезнаходження, колірне оформлення, зовнішній вигляд, вигляд і параметри шрифту.

6. Встановіть для командної кнопки **Command1**(Розрахунок) властивість **Default** в положення **True** для автоматичного виклику події **Click** для цієї кнопки при натисненні клавіші **Enter**. Аналогічно, властивість **Cancel** кнопки **Command2** (Вихід) встановіть в положення **True** для автоматичного виклику події **Click** для цієї кнопки при натисненні клавіші **Esc**.

7. Встановіть для об'єктів **Label2**, **Label3**, **Label4** і **Text2**, **Text3**, **Text4** властивість **Visible** (видимість поля) в положення **False**.

8. З подією **Click** командної кнопки C**ommand1** (Розрахунок) зв'яжіть такий код (зробіть подвійний клік на об'єкті **Command1,** у вікні, що розкрилося, введіть код):

```
Private Sub Command1_Click()
Text2.Text = 10 * Text1.Text \ 100
Text3.Text = 60 * Text1.Text \ 100
Text4.Text = Text1.Text - 10 * Text1.Text \ 100 - 60 * Text1.Text \ 100
Label2.Visible = True
Label3.Visible = True
Label4.Visible = True
Text2.Visible = True
Text3.Visible = True
Text4.Visible = True
End Sub
```

9**. З** подією **Click** командної кнопки **Command2(**Вихід**) з**в'яжіть такий код:

```
Private Sub Command2_Click()
End
End Sub
```

10. З подією **GotFocus** (одержання фокуса) об'єкта **Text1** зв'яжіть такий *код:*

```
Private Sub Text1_GotFocus()
    Text2.Visible = False
    Text3.Visible = False
    Text4.Visible = False
    Label2.Visible = False
    Label3.Visible = False
    Label4.Visible = False
End Sub
```

Ця подія виникає при переході до даного елементу й означає готовність поля **Text1** до приймання даних. Процедура обробки події робить невидимими останні текстові поля і мітки, прибираючи з екрана результати попередніх обчислень.

11. Запустіть додаток на виконання і виконайте обчислення для різних вихідних даних.

12. Збережіть проект з ім'ям **Lab_1**.

Завдання 2. Створіть проект для обчислення значень функцій

$$q = \frac{2x + \ln^2 x + \sqrt{x}}{a + mc + t};\ t = \frac{mb}{a + x};\ \ m = a + 2b^2;\ \ b = 4{,}3$$

Вихідні дані. x, a, b.

Результат. Вивести на формі q, t, m.

Обов'язкові умови. Введення вихідних даних організувати за допомогою об'єкту **TextBox**. Розрахунки і виведення даних повинні виконуватися при натисненні відповідної кнопки.

Порядок виконання роботи

13. Змініть назву проекту - **Lab_1_2**
14. Збережіть проект з новим ім'ям **Lab_1_2.**
15. Додайте текстове поле **Text5** і мітку **Label5.**
16. Змініть властивості об'єктів згідно з умовою завдання(див. приклад рис. 16).

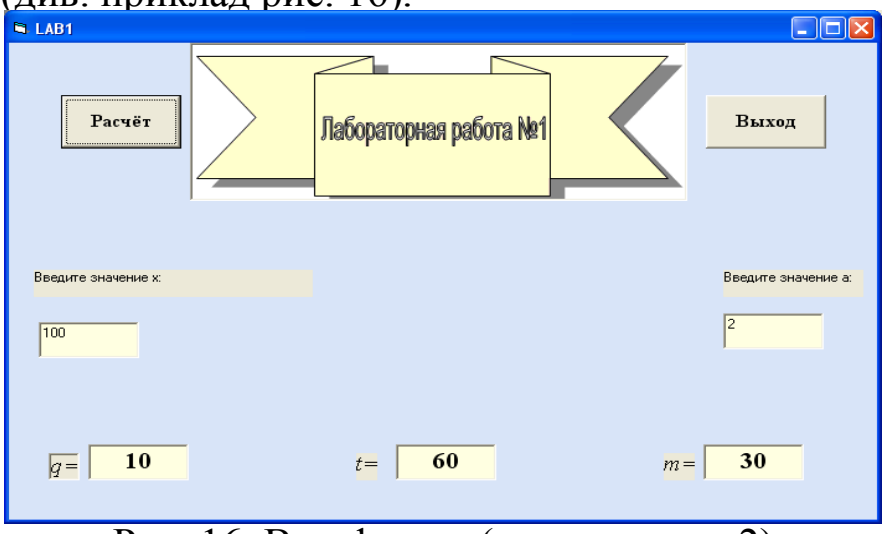


Рис. 16. Вид форми (для завдання 2)

17. Змініть властивість **MaxLength Text2**, **Text3**, **Text4**, встановивши значення 7(для обмеження кількості символів в значенні, що виводиться)

18. Змініть процедуру обробки події **Click** для **Command1** (Розрахунок):

```
Private Sub Command1_Click()
  b = 4.3
  x = Text1.Text
  a = Text5.Text
  m = a + 2 * b ^ 2
  t = m * b / a + x
  q = (2 * x + (Log(x)) ^ 2 + Sqr(x)) / (a + m * x + t)
  Text2.Text = q
  Text3.Text = t
  Text4.Text = m
  Label2.Visible = True
  Label3.Visible = True
  Label4.Visible = True
  Text2.Visible = True
  Text3.Visible = True
  Text4.Visible = True
End Sub
```

Завдання 3. Створіть проект для обчислення значення функції згідно з варіантом індивідуального завдання, вказаного в розд.4, лабораторної роботи 2.

РОБОТА 18
ПРОЕКТУВАННЯ ДОДАТКА VB6 З ВИКОРИСТАННЯМ ОПЕРАТОРІВ УМОВНОГО ПЕРЕХОДУ

Завдання 1. Створіть проект, в якому розрахуйте можливу кількість локомотивів, що знаходяться в ремонті, залежно від періоду експлуатації. Врахувати, що в зимовий період одночасно в експлуатації повинно знаходитися 40% рухомого складу, резерв –10%, у весняно-літній період – одночасно в експлуатації знаходиться в 1,5 рази більше локомотивів, ніж в літній період.

Вихідні дані. Ввести за допомогою розташованих на формі об'єктів **TextBox** загальну кількість локомотивів.

Результат. Вивести на формі кількість локомотивів, які експлуатуються, резерв і кількість локомотивів у ремонті.

Обов'язкові умови. Розрахунки і виведення даних повинні відбуватися при натисненні відповідної кнопки. Результати повинні виводитися на окремій формі.

Порядок виконання роботи

1. Увійдіть в середовище Visual Basic. Відкрийте збережений проект **Lab_1**. Перейменуйте форму **Form1** замініть на **frmIshDan**, проект назвіть **Lab2.** Збережіть проект під ім'ям **Lab_2**. Виріжте з форми об'єкти **Label2, Label3, Label4 і Text2, Text3, Text4.**

2. Змініть розміщення об'єктів на формі і додайте контейнер Frame, в якому розмістіть два об'єкти **OptionButton**

3. Змініть властивість **Caption** об'єктів **Frame, Option1 і Option2,** (згідно з рис.17)

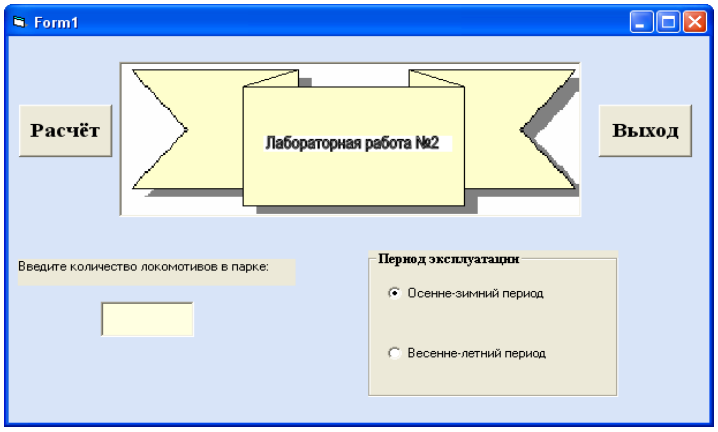


Рис. 17. Перемикач, розміщений в об’єкті-контейнері

4. Додайте до проекту ще одну форму з назвою **Frmrez.** За допомогою вікна розміщення форм **Form Layout** розмістіть форми так, щоб вони не перекривали одна одну. Вставте у форму **Frmrez** об'єкти **Label2, Label3, Label4 і Text2, Text3, Text4.** Відредагуйте властивості форми та об'єктів, розміщених на ній, на свій розсуд (колір фону, розміри, шрифт, розташування і т. д.)

5. Відкрийте редактор коду і для форми **frmIshDan** змініть процедуру **Private Sub Command1_click()** таким чином:

```
Private Sub Command1_click()
  If Option1.Value = True Then
frmRez.Expluat.Text = 40 * frmIshDan.Text1.Text \ 100
  frmRez.Rezerv.Text = 10 * frmIshDan.Text1.Text \ 100
  frmRez.Remont.Text = frmIshDan.Text1.Text-frmRez.Expluat.Text - frmRez.Rezerv.Text
Else
   frmRez.Expluat.Text = 1.5 * 40 * frmIshDan.Text1.Text \ 100
   frmRez.Rezerv.Text = 10 * frmIshDan.Text1.Text \ 100
  frmRez.Remont.Text = frmIshDan.Text1.Text - frmRez.Expluat.Text -frmRez.Rezerv.Text
 End If
   frmRez.Visible = True
End Sub
```

6. Пов'яжіть з подією **LostFocus** (втрата фокуса) поля **Text1** такий код:

```
Private Sub Text1_LostFocus()
   If Not IsNumeric(Text1) Then
   MsgBox "Невірне значення: не цифра"
   Text1.SetFocus
   End If
End Sub
```

Ця подія настає після втрати фокуса елементом управління, тобто після передачі фокуса іншому елементу по завершенню введення або корегування даних. Описана вище процедура за допомогою вбудованої функції **IsNumeric()** перевіряє, чи можна введене значення привести до числового типу. При негативному результаті за допомогою функції **MsgBox** виводиться повідомлення про помилку і фокус повертається до поля **Text1** для виправлення помилки.

7. Дослідіть виконання програми з різними даними. Результати додайте до звіту.

Завдання 2: Створіть проект для обчислення значення функції

$$y=\begin{cases} cx^3+a, \text{ якщо } x=5 \text{ і } b+c<12 \\ \ln(x-a), \text{ якщо } c\le x<b \text{ і } x\neq 5 \\ b+3c, \text{ в інших випадках} \end{cases}$$

Вихідні дані. x, a, b, c.

Результат. Вивести на формі в, R.

Обов'язкові умови. Введення вихідних даних організувати за допомогою функції **InputBox**. Розрахунки і виведення даних повинні здійснюватися при натисненні відповідної кнопки. Результати повинні виводитися на окремій формі.

Порядок виконання завдання

1. Створіть новий проект (**Standard Exe**).

Назвіть Ваш проект - **Lab_2_2.**

2. Змініть назву форми**: frmVichFunc**.

3. Розмістіть на формі об'єкти, відредагуйте властивості форми і об'єктів відповідно до рис. 18.

4. Для об'єктів **Label1** і **Text1** властивість **Visible** встановіть в в положення **False.**

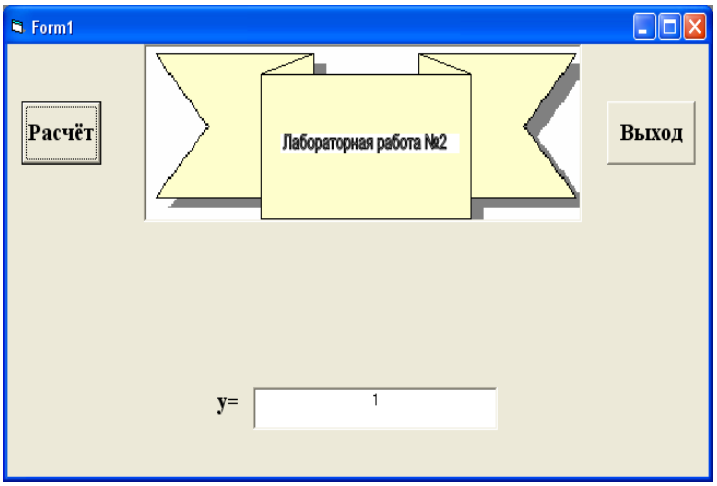


Рис. 18. Приклад форми для завдання 2.

5. Для кнопки **Command2**(Вихід) створіть процедуру:

```
Private Sub Command2_click()
 End
End Sub
```

6. Для кнопки **Command1**(Розрахунок) створіть процедуру:

```
Private Sub Command1_Click()
    Dim a, x, c, b As Integer
    Dim y As Single
    a = InputBox("Уведіть a", , 3000, 3000)
    b = InputBox("Уведіть b", 1, 5000, 3000)
    c = InputBox("Уведіть c", 1, 5000, 3000)
    x = InputBox("Уведіть x", 1, 5000, 3000)
    If x = 5 And b + c < 12 Then
    y = cx ^ 3 + a
    ElseIf c <= x And x < b And x <> 5 Then
    y = Log(x - a)
    Else: y = b + 3 * c
    End If
    Text1.Text = y
    Label1.Visible = True
    Text1.Visible = True
End Sub
```

Завдання 3. Створіть проект для обчислення значення функції згідно з варіантом індивідуального завдання, вказаного в розд.4, лабораторна робота 3.

РОБОТА 19

ПРОЕКТУВАННЯ ДОДАТКА VB6 З ВИКОРИСТАННЯМ ОПЕРАТОРІВ ЦИКЛУ

Завдання 1 У проекті передбачити можливість розрахунку середньої кількості локомотивів у ремонті за заданий період.

Вихідні дані. Ввести за допомогою розташованих на формі об'єктів **Textbox** загальну кількість локомотивів.

Результат. Вивести на формі кількість локомотивів, які експлуатуються, резерв і кількість локомотивів у ремонті. Результати розрахунків записати у файл. Обчислити середню кількість локомотивів у ремонті за заданий період.

Обов'язкові умови. Розрахунки і виведення даних повинні здійснюватися при натисненні відповідної кнопки. Середню кількість локомотивів у ремонті за заданий період вивести на окремій формі.

Порядок виконання роботи:

1. Увійдіть в середовище Visual Basic. Відкрийте збережений проект **Lab_2**. Змініть проект і додайте об'єкт **ComboBox** (список, що розкривається).(рис. 19)

Змініть властивість **Name** об’єкта **ComboBox** на **Combo_month.**

Щоб розмістити значення в списку під час розроблення, виконайте такі дії:

1.1. Виділіть властивість **List** (Список).

1.2. У правому стовпці властивості з'явиться кнопка, що містить направлену вниз стрілку. Натисніть цю кнопку. Відкриється список, що дозволяє вводити значення.

1.3. Введіть перше значення списку(січень).

1.4. Для переходу на новий рядок списку натисніть комбінацію клавіш <**Ctrl**>+<**Enter**>.

1.5. Введіть наступне значення списку.

1.6. Повторюючи пункти 1.4 і 1.5, сформуйте весь список.

Щоб при появі форми на екрані в списку за попереднім визначенням було виділене певне значення використовується властивість **Listindex**.

```
Private Sub Form Load()
   Combo_month.ListIndex = 0
End Sub
```

У полі списку відображуватиметься перший елемент списку (січень). Якщо для списку, що розкривається, не встановлене використовуване за попереднім визначенням значення, то при появі його на екрані в полі, призначеному для введення

значення списку, відображається текст **Combol**, що задається властивістю **Text** і що є ім'ям об'єкта. Якщо ви хочете, щоб це поле при появі списку на екрані було порожнім або містило заданий вами текст, виділіть властивість **Text** і в правому стовпці видаліть інформацію, залишивши поле порожнім, або введіть необхідний текст, відповідно до рис. 19.

Рис. 19. Зовнішній вигляд форми (**frmIshDan**)

2. Змініть код проекту згідно з приведеним нижче:

```
Private Sub Form Load()
    Combo_month.Listindex = 0
End Sub

Private Sub run_Click()
      If Combo_month.ListIndex <= 3 Or Combo_month.ListIndex >= 9 And
      combo_month.ListIndex <= 11 Then
     kol_expl.Text = 40 * all.Text \ 100
     kol_rez.Text = 10 * all.Text \ 100
     kol_rem.Text = all.Text - kol_expl.Text - kol_rez.Text
     Else
     kol_expl.Text = 1.5 * 40 * all.Text \ 100
     kol_rez.Text = 10 * all.Text \ 100
     kol_rem.Text = all.Text - kol_expl.Text - kol_rez.Text
     End If
     a = kol_rem.Text
    Open "dan.txt" For Append As #1
     Write #1, a
     Close #1
End Sub

Private Sub exit_Click()
   Form2.Visible = True
End Sub
```

3. Спроектуйте форму (**Frmrez**) згідно з рис. 20.

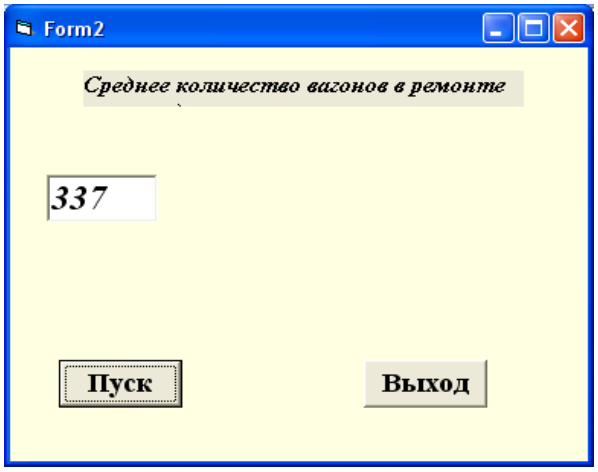


Рис. 20. Зовнішній вигляд форми (**Frmrez**)

4. Додайте такий код для цієї форми:

```
Private Sub Command1_Click()
    Open "dan.txt" For Input As #1
    k = 0
    s = 0
    Do
    Input #1, a
    s = s + a
    k = k + 1
     Loop While Not EOF(1)
     Close #1
     Text1.Visible = True
     Text1.Text = s \ k
End Sub

Private Sub Command2_Click()
End
End Sub
```

5. Запустіть проект на виконання. Перевірте роботу програми з різними наборами даних.

Завдання 2: Створіть проект для обчислення значення функції згідно з варіантом індивідуального завдання, вказаного в розд.4, лабораторна робота 4.

4. ТЕСТИ

4.1. ВАРІАНТИ ЗАВДАНЬ ЗА ТЕМАТИЧНИМИ БЛОКАМИ

Контрольні запитання з теоретичного матеріалу

QBasic

1 Типи даних.
2 Арифметичні дані.
3 Символьні дані.
4 Арифметичні вирази.
5 Оператор присвоювання.
6 Оператор введення (INPUT).
7 Оператори READ, DATA .
8 Оператор виведення (PRINT).
9 Оператор виведення (PRINT USING)
10 Оператор безумовного переходу (GOTO)
11 Оператор умовного переходу (IF...THEN).
12 Оператор вибору (SELECT CASE).
13 Оператори циклу (FOR NEXT)
14 Оператор циклу (DO LOOP).
15 Оператор циклу з передумовою.
16 Оператор циклу з післяумовою.
17 Оператори для завдання значень змінних.
18 Оператори циклу.
19 Пріоритет операцій .
20 Математичні функції.
21 Логічні вирази .
22 Арифметичні константи.
23 Оператор DIM .
24 Введення елементів масивів.
25 Задавання типу змінної.

Visual Basic

1. Як відкрити вікно інтегрованого середовища розроблення програм VB6?
2. Які основні елементи інтегрованого середовища VB6?
3. Назвіть основні пункти меню інтегрованого середовища VB6.
4. Що є основою програм на VB6?
5. Що таке модуль?
6. Як поділяються модулі в VB6?
7. Що являють собою стандартні модулі VB6? Як вони створюються?
8. Як створюються модулі форм і звітів VB6?
9. Що таке процедура? Який синтаксис процедур?
10. Як здійснюється виклик процедур? Що при цьому відбувається?
11. Як здійснюється обмін даними між процедурами і головною програмою?
12. Які аргументи використовуються при описі процедур?

13. Який синтаксис функції? У чому її відмінність від процедури?
14. Яке призначення розділу описів?
15. Як використовуються змінні в процедурі?
16. Яка область видимості змінних, описаних у модулі, у процедурі або функції?
17. Як описати загальну змінну, доступну для всіх процедур?
18. За якими правилами можна задати ім'я змінної в VB6?
19. Перерахуйте і дайте стислу характеристику основних типів даних в VB6?
20. Які особливості типу Variant?
21. Як заборонити неявний опис змінних в VB6?
22. Перелічіть і дайте стислу характеристику операторам керування виконанням програми. Наведіть приклади їхнього використання.
23. Як можна ввести дані в програму?
24. За допомогою яких функцій можна перетворити строкове значення в число?
25. Як можна вивести на екран результат обчислень?

Лабораторна робота 1

Завдання 1. *Вказати правильно записані мовою Basic ідентифікатори та їх тип.*

Завдання.2. *Записати константи у природній та експоненціальній формах.*

Завдання 3. *Написати програму введення та виведення даних* (табл. 16)**.**

Таблиця 16

Номер вар.	Завдання 1	Завдання 2	Завдання 3
1	2	3	4
1	SK 105! R+61 8K51 -BS02	35,62 0,32 8,831 36,358	a=0,61 b=3,645 c=18 d=0,051
2	5R1 *B1 2ARC ARC&	21,001 0,5 211,05 100	a=12345 b=-21,2 c=345 d=0,00003
3	4C ‘ABC’ XYZ% 1234	-195 341,5 -0,002 -1,00001	a=0,1 b=2,22 c=-34567 d=25,52
4	A.B# X00B00! P1112A1 -AB12	123456 -123645 0,00009 -1111	a=-0,00001 b=0,5610 c=123 d=-1313.1
5	S25,1R 10AB R12345R# R00000%	123,123 2,3333 -0,1 4231,5	a=0,110 b=-100 c=14.5 d=-3.14

Продовження таблиці 16

1	2	3	4
6	**444** **A-R#** **44B** **(XYZ)**	**21,021** **-1765** **5** **0,0002**	**a=3,14159** **b=-0,00001** **c=49** **d=0**
7	**B.A%** **(PP)X!** **XXXXXX&** **-PK**	**444,444** **-100000** **-777.1** **12345**	**a=0,05** **b=0,9110** **c=0,1234567810** **d=159**
8	**%C12-1&** **-R&** **R0.2X%** **B000!!**	**-21,4** **345** **0,00028** **-100000**	**a=1,0101** **b=345** **c=0,0001]** **d=12,98**
9	**SSSSPS%** **S111111&** **-X4** **'ABC'$**	**1000000** **99.99** **-1** **678**	**a=-98765** **b=548** **c=341** **d=10000.1**
10	**+BS00.12%** **+PR1#** **X011111!%** **AAAAAA!**	**-2123** **549,26** **744** **0,744**	**a=0,0110** **b=23** **c=-0,23** **d=192837**
11	**STON.4&** **DOW!%** **-X12** **4R21B$**	**-0.1** **12,43** **314159** **412**	**a=8,5910** **b=-0,510** **c=14** **d=543**
12	**XФ5&** **2PX%!** **-249A** **DOOR%**	**0** **546** **751,001** **-4,4**	**a=3,510** **b=800** **c=947,74** **d=-34**
13	**FFFFFF@!** **RIII11#** **R19S&!** **+N10**	**213** **1000000** **-1000000** **56,05**	**a=0,0001** **b=-0,0026** **c=1000000** **d=425,7**
14	**+ORT** **Ж14AS** **0A1$** **Z!**	**34,345** **-100000** **-0,0055** **753,1**	**a=200** **b=4000.5** **c=-4000** **d=0.75463219**
15	**'NNK'** **CHNAR&** **Ф1!** **FUT.B%**	**444,44** **-1122** **-99999** **4,9**	**a=152,14** **b=12** **c=-100000** **d=-239,01**

Лабораторна робота 2

Завдання. *Скласти схему алгоритму та програму.*

Задача 1

$f = z^3 + z + 1;\ k = z - 35$

$z = a^2 \cos(x+1) + 2\sin(y^2 + 5)$

$x = 0{,}25 + t + \cos^3(0{,}25 + t)$

t – будь-яке

$a = 14S + 1$

S = 9,6

y – будь-яке

Задача 2

$p = n + f^2 + f$

$y = p^2 + c + m^2 f$

$f = 3c^3 + 5 + \cos^3(3c^3 + 5) + \sin(3c^3 + 5)$

$n = m^7 + \sqrt{m+1}$

$m = db + \dfrac{\sqrt{b^2 + 7}}{15}$

d, b, c – будь-які, a=16

Задача 3

$x = 0{,}05 + \sin(a^2 yz) - 6 - z^5$

$y = \sin(a^2 bz) + |a^2 bz| + \dfrac{a^2 bz}{7}$

b=0,02; a – будь-яке

$z = 100 - \dfrac{189}{|b - a|}$

$z = 3p + 8p^2 + p^6$

p = 0,6

Задача 4

$S = Vt;\ k = at + 5$

$a = \sqrt[3]{S + 25} + \sin(S + 25) + |S^5 - 1 + S|$

t – будь-яке;

$V = \dfrac{u^2 + w^2}{5}$

$u = t^7 + 3;\ w = 3{,}41$

Задача 5

$S = at^2 + 1$

$p = 6{,}2;$

$z = b\sin x + \ln|b\sin x|$

$x = y^3 + py^2 + Sy$

t, a – будь-яке

$f = \sqrt{\sin S + xa} + |p - 1 + \sin S|$

$y = a^5 - a + 7;$

$b = \sin(S)$

Задача 6

$n = a^3 + \sqrt{a^3 - 1}$

$C = 0{,}5f^3 + f + 1;$

$Q = S^3 + n - 5$

$b = 25 - a;$

f –будь-яке; a=-3,1

$r = n + a + \sin(n + a) - \sqrt{n + a}$

$S = C + rb + r^2 a$

Задача 7

$m = 3{,}2a + 6{,}45b - c$

$k = m^3 + a - b$

a – будь-яке

$t = a^2 + mb - k$

$c = -6{,}66$

$b = a^3 + a - 1 + \ln|a^3 + a - 1|$

$s = m + kt$

Задача 8

a, b, c –будь-які; d=0,35

$n = m^3 + \sqrt{am} + \sin(am)$

$m = b - 0{,}65d$

$z = 3(c^2 + 1) + |c^2 + 1|$

$y = x + c - mz$

$x = n + z$

Задача 9

$b = 10s + s^2 + 1$

$x = 0{,}7 + t - \cos(0{,}7 + t)$

$s = -3{,}4;\ m = y + z$

t –будь-яке

$Q = z + z^5 + 3;\ z = b - 4$

$y = b^3 \cos(x - 1) - 4\sin(b + 5)$

Задача 10

$d = f^2 + a + b$

$f = 30a + 22b + c^5$

$t = a^2 + f^3 + d$

c=105

$b = a^7 + \ln(a + 1) + \sin(a^7 + \ln(a + 1))$

$Q = f + dt;$

a – будь-яке

Задача 11

x – буд- яке; a – будь-яке

$n = am^2 + 47a^2 m^5$

$m = xa - \cos^3(xa)$

t=-3,33

$p = (n + m) + z^3 + \sin(m)$

$z = t^3 + x^2 + a;\ s = p^3 + 5$

Задача 12

A=0,654

d, t, c – будь-які

$z = d^3 + 6$

$y = z - d + \sqrt{d^2 + 1} + d^2 + 1$

$x = z - A$

$u = |tc| - \cos y + l$

$l = \sin Atc$

Задача 13

$i = 0{,}8j + 25k$

$p = 2\sin(s+1) + \sqrt{s+1} + \ln(s+1)$

$S = abc$

a, b – будь-які

j*=3,45*

$k = 12j - j^7 + 5$

*c=6,31-****j***

$f = 100 + \dfrac{14}{i+j+k}$

Задача14

$p = z^3 + az^2 + bz + c$

a – будь-яке

$b = 0{,}12Q - 1$

f=154

$z = a - t^5 + \sin|a - t^5|$

$c = 0{,}34f - f^3 + f^5$

$Q = \sin(f+1) + t$

$t = a + \sqrt{|a+f|}$

Задача 15

$a = \dfrac{b^3 + x^{z+k}}{0.63y^2 + 1}$

$x = 2z^5 + y$

$f = \sin(c)$

$y = z + \sqrt[3]{f}$

c – будь-яке

k=3; b=4; z=14

Лабораторна робота 3

Прості логічні вирази *Скласти схему алгоритму та програму (табл. 17).*

Таблиця 17

Номер варіанта	*ЗАВДАННЯ*	
1	2	
1	m=	x-1, якщо x<1 ax+b, якщо x=1, при a=12,4; b=56,1; x –довільне значення 1+b/a, якщо x>1
2	m=	x+1, якщо x<1 a-x+b, якщо x=1, при a=0,001; b=5,1; x –довільне значення x-(b/a), якщо x>1
3	f=	1-cos x-y, якщо y<0 (ay+b)/2, якщо y=0, при a=0,2; b=0,01; x,y –довільні значення a+1, якщо y>0
4	m=	1-a cos x, якщо x<1 ax+b, якщо x=1, при a=0,001; b=5,1; x –довільне значення x+(b/a), якщо x>1
5	x=	log(ay+1), якщо y<2 (ay+b)/2, якщо y=0, при a=0,2; b=0,01; y –довільне значення a+1, якщо y>0
6	m=	x+1, якщо x<1 ax+b, якщо x=1, при a=0,001; b=5,1; x –довільне значення x+(b/a), якщо x>1
7	w=	b-yx, якщо y>1 sin(b-y), якщо y=1 (x-y)/2, якщо y<1, при b=0,15; x=2, y –довільне значення

Продовження таблиці 17

1	2
8	f= { c-ba, якщо a>0 a-((b/2)+1), якщо a=0, при b=1,2;c=1,71; a –довільне значення a-1, якщо a<0
9	f= { 0,5y-1, якщо y=1 (1-y)/a, якщо y>1, при a=18,12; y –довільне значення (1+y), якщо y<1
10	f= { (1-x)/2, якщо x>3 sin x-2, якщо x=3 (1-x)/(a-b), якщо x<3, при a=5,5; b=1,2; x –довільне значення
11	k= { 1-xy, якщо y<3 1+xy, якщо y=3, при x=0, 812; y –довільне значення 1-2xy, якщо y>3
12	y= { (x+2)/3, якщо x<2 x-a, якщо x=2 (x-2)/(a-1), якщо x>2, при a=7,21 x –довільне значення
13	s= { 1-y, якщо y=1 (1+y)/a, якщо y<1, при a=0,412; y –довільне значення 1+3,5y, якщо y>1
14	g= { (1-x), якщо x<0 (1+2x)/(a-2), якщо x>0, при a=6,31; x –довільне значення 1+x, якщо x=0
15	f= { ((a-b)/2+x), якщо x>1 x log(a+b), якщо x=1 ((a+b)x)/a, якщо x<1, при a=0,7; b=0,31 x –довільне значення

Складні логічні вирази. *Скласти схему алгоритму та програму*
Завдання 1 Дані 3 числа A,B,C. Визначити та надрукувати:
Задача 1. Числа, що кратні двом.
Задача 2. Числа, що кратні трьом.
Задача 3. Парні числа.
Задача 4. Непарні числа.
Задача 5. Кількість додатніх чисел.
Задача 6. Кількість парних чисел.
Задача 7 Кількість чисел > 1.
Задача 8. Кількість чисел, розташованих в діапазоні (–2,+15).
Задача 9. Кількість чисел < 1.
Задача 10. Кількість чисел =0.
Задача 11. Числа, що кратні п‘яти.
Задача 12. Чи можна з відрізків A,B,C побудувати трикутник.
Задача 13. Чи складають вони арифметичну прогресію.
Задача 14. Чи виконується умова A>B>C.
Задача 15. Чи виконується умова A<B<C.

Завдання 2 *Скласти схему алгоритму та програму(табл. 18)*

Таблиця 18

Номер вар.	*Завдання*
1	2
1	$z=\begin{cases}\sin(x+y), & \text{якщо } x>1 \text{ або } y>1;\\ \lvert x+y\rvert, & \text{якщо } x<-1 \text{ і } x>y;\\ e^{x-y}, & \text{якщо } 0\le x\le 1,\\ \cos(x+1), & \text{в інших випадках}\end{cases}$ x,y - довільні значення
2	$y=\begin{cases}\lvert x^3\rvert, & \text{якщо } x<ab \text{ і } x<-2;\\ \sin(x^a+b), & \text{якщо } x>ab \text{ або } x>2;\\ a+b+1, & \text{якщо } 0<x<1 \text{ і } x=ab;\\ x-(a+b)^2, & \text{в інших випадках}.\end{cases}$ x,a,b - довільні значення
3	$f=\begin{cases}c+x, & \text{якщо } 1\le x\le 3;\\ c-x, & \text{якщо } 3<x\le 10 \text{ і } c>0;\\ \lvert c\rvert^x, & \text{якщо } x>10;\\ x, & \text{в інших випадках}.\end{cases}$ c,x - довільні значення
4	$w=\begin{cases}e^v, & \text{якщо } v>x \text{ або } v>y;\\ (x+y), & \text{якщо } v=10 \text{ і } y>x>15;\\ v^{x+y}, & \text{якщо } v=x;\\ 1, & \text{в інших випадках}.\end{cases}$ v,x,y - довільні значення
5	$t=\begin{cases}s+2z, & \text{якщо } s+z=2{,}8 \text{ або } s+z=1;\\ s-z^2, & \text{якщо } s+z>5 \text{ і } z>0;\\ \lvert s+z\rvert, & \text{якщо } 3<s+z\le 5;\\ a+s, & \text{в інших випадках}.\end{cases}$ a,s,z - довільні значення
6	$y=\begin{cases}a+b+c, & \text{якщо } a+b>c;\\ (a+b)-c, & \text{якщо } a+b<c \text{ і } c>5;\\ \lvert\sin(b)\rvert, & \text{якщо } a+b=c;\\ a^{bc} & \text{в інших випадках}.\end{cases}$ a,b,c - довільні значення
7	$z=\begin{cases}1, & \text{якщо } y<0;\\ 2, & \text{якщо } 0\le y<5;\\ 3, & \text{якщо } 5\le y<10;\\ 4, & \text{якщо } y\ge 10.\end{cases}$ y - довільне значення
8	$s=\begin{cases}t-1, & \text{якщо } 0<zt<5;\\ z, & \text{якщо } zt>10 \text{ або } zt<-3;\\ t, & \text{якщо } zt=0;\\ 0, & \text{в інших випадках}.\end{cases}$ z,t- довільні значення

1	2
9	$m=\begin{cases}\sin^2(x)+\cos(x), & якщо\ x>0;\\ \dfrac{x+y}{a^2+2}+\sin\left(\dfrac{x+y}{a^2+2}\right), & якщо\ x\le 0\ \ i\ \ a>0;\\ x+\lvert a-y\rvert, & в\ інших\ випадках.\end{cases}$ x,y,a, - довільні значення
10	$n=\begin{cases}\lvert m+c\rvert, & якщо\ \ m>10\ \ або\ \ m=4;\\ m-c, & якщо\ \ m<3;\\ m, & якщо\ \ 5<m<8;\\ mc, & в\ інших\ випадках.\end{cases}$ m,c - довільні значення
11	$. x=\begin{cases}t+q, & якщо\ \ t>10\ \ i\ \ t=q;\\ t+1, & якщо\ \ t\le -10;\\ t, & якщо\ \ t=5;\\ 1+tq & в\ інших\ випадках.\end{cases}$ t,q - довільні значення
12	$. k=\begin{cases}x+y^2+2, & якщо\ \ x=y+2\ ;\\ x+y+2, & якщо\ \ x>y+2\ \ i\ \ y=3;\\ \sin(x)+\sin(2), & якщо\ \ 0<x<y+2;\\ x+\ln\lvert y+2y\rvert & віншихвипадках.\end{cases}$ x,y - довільні значення
13	$z=\begin{cases}2k, & якщо\ \ k<2;\\ 3k, & якщо\ \ 2\le k\le 6;\\ 4k, & якщо\ \ 6<k<10;\\ 5k, & якщо\ k\ge 10.\end{cases}$ k – довільне значення
14	$p=\begin{cases}t+a, & якщо\ \ t>0\ \ \ або\ \ \ a>0;\\ t-a, & якщо\ \ t<0\ \ \ i\ \ \ t<0;\\ \lvert a\rvert & в\ інших\ випадках.\end{cases}$ a,t - довільні значення
15	$g=\begin{cases}\ln(a^2+1), & якщо\ \ 15\le a<20;\\ 2a+7, & якщо\ \ 10\le a<15;\\ 1-ab, & якщо\ \ 8<a<10;\\ \dfrac{4a}{7(a+b)^2+5}, & якщо\ \ 0\le a<8\ \ i\ \ a>b;\\ a+b & в\ \ інших\ \ випадках.\end{cases}$ a,b - довільні значення

Лабораторна робота 4

Завдання. *Протабулювати функцію та визначити суму обчислених значень.*

Таблиця 19

Варіант	Функція	Вихідні дані				
		a	**в**	**xn**	**xk**	**h**
1	$Y=\frac{arctgbx}{1+\sin^2 x}$	-		1.35	6.5	0.8
2	$Y=5\sqrt{\frac{a+bx}{\ln^2 x}}$	19.6	7.8	14.6	34.8	6
3	$Y=\frac{a\ln^2 x}{b+\sqrt{x}}$	1.38	-1.26	60	100	10
4	$Y=\frac{\sin^2 x}{\sqrt{x}+bx}$	-	1.68	1.2	2.4	0.2
5	$Y=\frac{\ln^2(x-b)}{a\sqrt{x}}$	0.36	5.5	10	50	6
6	$Y=\frac{e^{ax}+b}{1+\cos^2 x}$	0.9	1.85	0	1.2	0.15
7	$Y=\frac{a+\sqrt[3]{x}}{\sin^2 bx}$	1.24	0.67	10.2	12.4	0.45
8	$Y=\frac{a\sqrt{x}-bx}{\ln^3 x}$	2.8	0.45	40	60	4.5
9	$Y=\frac{\sqrt{ax}-b}{tg^3 x}$	20.2	7.65	3.5	4	0.1
10	$Y=e^{-x^2}\frac{a+bx}{\ln^2 x}$	4.6	2.5	0.75	1.8	0.3
11	$Y=\frac{tg^2 ax-b}{e^{ax}}$	0.55	0.78	4.2	5.8	0.25
12	$Y=\frac{arctgbx}{1+\sqrt[5]{ax}}$	7.38	0.3	9	12	0.35
13	$Y=\frac{\sin^3 ax}{ax+b}$	0.28	1.35	1.2	7.5	0.5
14	$Y=\frac{e^{-bx}}{b+\cos^3 ax}$	0.9	0.66	2.3	8.9	1.3
15	$Y=\frac{\ln^2\sqrt{x}}{a\sqrt{x}}$	0.85	-	17.2	24.6	2

Лабораторна робота 5

Завдання. *Скласти схему алгоритму та програму обчислення.*

Задача 1. Дано натуральне число N та дійсне число f.

$$P=\prod_{i=1}^{N}\frac{1}{\sum_{k=0}^{1}(f+k)}.$$

Задача 2. Дано натуральне число N.

$$S=\sum_{k=1}^{N}(-1)^{k-1}*\prod_{m=1}^{2k}\cos\frac{m+1}{2k}\quad .$$

Задача 3. Дано натуральне число N.

$$S=\sum_{k=1}^{N}\frac{\sum_{i=1}^{N}\sin(0.01*k*i)}{k!}\quad .$$

Задача 4. Дано натуральне число N.

$$S=\sum_{i=1}^{N}\prod_{j=1}^{i}\frac{j!}{i!}.$$

Задача 5. Дано натуральне число N та дійсне число x.

$$S=\sum_{i=1}^{N}\prod_{k=1}^{N}\frac{i+x}{k}.$$

Задача 6. Дано натуральне число N та дійсне число x.

$$S=\sum_{i=1}^{N}(-1)^{i+1}\frac{x^{2i}}{i!}.$$

Задача 7. Дано натуральне число N.

$$S=\sum_{k=1}^{N}\prod_{m=1}^{2k}\sin\frac{m\pi}{2k+1}.$$

Задача 8. Дано: l,n – будь-які числа.

$$S=\prod_{a=1}^{5}\left(\frac{\sum_{i=1}^{l}\prod_{j=1}^{n}(i+j)^{a}}{a!}\right).$$

Задача 9. Дано натуральне число N.

$$F=\prod_{i=1}^{12}\frac{\sum_{j=1}^{i}\sum_{k=1}^{j}(k+i)^{2}}{(i+5)!}.$$

Задача 10.

$$T=\frac{\prod_{i=1}^{30}\sum_{j=1}^{l}(b+c)^{j+i}}{(m+l)!-l!}.$$

Задача 11.

$$F=\frac{\sum_{i=1}^{10}\prod_{j=1}^{3}(a+b)^{i=j}}{a!}.$$

Задача 12.

$$S=\prod_{c=1}^{5}\left(\frac{\sum_{i=1}^{m}\prod_{j=1}^{n}(i+j)^{c}}{c!}\right).$$

Задача 13.

$$F=\frac{\prod_{i=1}^{30}\sum_{j=1}^{n}(b+c)^{i+j}}{(m+n)!-n!}.$$

Задача 14. Дано натуральне число N та дійсне число x.

$$S=\sum_{i=0}^{n}\frac{x^{2i+1}}{(2i+1)!}.$$

Задача 15. Дано натуральне число N та дійсне число x.

$$S=\sum_{i=0}^{n}(-1)^i\frac{x^{2i}}{(2i)!}.$$

Лабораторна робота 6

Завдання. *Скласти схему алгоритму та програму.*
З заданою точністю **EPS**

*Задача 1.*Обчислити *ln(1+x)* за наближеною формулою

$$\ln(1+x)=x-\frac{x^2}{2}+\frac{x^3}{3}-\ldots+(-1)^{n-1}\frac{x^n}{n}.$$

Задача 2. Обчислити інтегральну функцію Φ(x) за наближеною формулою (x>0)

$$\Phi(x)=x\sqrt{\frac{2}{\pi}}\left(1-\frac{(x/2)^2}{1!3}+\frac{(x/2)^4}{2!5}-\frac{(x/2)^6}{3!7}+\ldots\right).$$

Задача 3. Обчислити ln2 за наближеною формулою

$$\ln 2=1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots\pm\frac{1}{n}.$$

Задача 4. Обчислити константу e за наближеною формулою

$e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\ldots..+\frac{1}{n!}$, де n – натуральне число.

Задача 5. Обчислити *arctg(x)* за наближеною формулою.

$$arctg(x)=x-\frac{x^3}{3}+\frac{x^5}{5}-\ldots+(-1)^{m-1}\frac{x^{2m-1}}{2m-1}.$$

Задача 6. Обчислити гіперболічний косінус за наближеною формулою $chx=1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}+\ldots+\frac{x^{2n}}{(2n)!}$, де $|x|<\infty$.

Задача 7. Обчислити гіперболіний сінус за наближеною формулою

$\operatorname{sh} x=x+\frac{x^3}{3!}+\frac{x^5}{5!}+\frac{x^7}{7!}+\ldots+\frac{x^{2n+1}}{(2n+1)!}$, де $|x|<\infty$.

Задача 8. Обчислити lnx за наближеною формулою

$$\ln x=2\left[\frac{(x-1)}{x+1}+\frac{(x-1)^3}{3(x+1)^3}+\frac{(x-1)^5}{5(x+1)^5}+\ldots+\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}\right],$$

де n – натуральне число.

Задача 9. Обчислити y=a*a , где $a=1+\frac{1}{2}+\frac{1}{4}+\ldots+\frac{1}{2^n}$,

де n – натуральне число.

Задача 10. Обчислити a^x за наближеною формулою

$$a^x = e^{x\ln a} = 1 + \frac{x\ln a}{1!} + \frac{(x\ln a)^2}{2!} + \frac{(x\ln a)^3}{3!} + \ldots$$

Задача 11. Знайти суму ряду

$$\frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \frac{(x-1)^3}{3x^3} + \ldots + \frac{(x-1)^n}{nx^n} + \ldots$$

Задача 12. Обчислити число π з точністю 10^{-6}, користуючись рядом Грегорі

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots + (-1)^{n+1} * 1/n - 1 + \ldots$$

Задача 13. Написати програму обчислення e^x за формулою

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots$$

Задача 14. Обчислити sin x за наближеною формулою (в радіанах)

$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$, де n - натуральне число.

Задача 15. Обчислити cos x за наближеною формулою (в радіанах)

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + (-1)^n \frac{x^{2n}}{(2n)!}.$$

Лабораторна робота 7

Завдання. *Скласти схему алгоритму та програму.*

Задача 1. Яке за порядком значення функції є мінімальним?

Y= - a sin²d $d \in [m, l]$ $h_d = 0.5$.

Задача 2. Знайти різницю між порядковими номерами максимуму та мінімуму функції

Z=a cos²x $x \in [3, 15]$ $\Delta x = 0{,}2$ *a=2.*

Задача 3. . Визначити, чи є сьоме значення функції *y=cos³x-ln|x-10|* максимальним $x \in [1, 10]$ $\Delta x = 1$; вивести на екран відповідне повідомлення

Задача 4. Знайти порядковий номер мінімального значення та аргумент при максимальному значенні функції

T=a tgb $b \in [b_1, b_2]$ $\Delta b = h$ *a, b1, b2, h* – будь-які числа

Задача 5. Знайти значення мінімуму та відповідного аргументу функції

$Z = \cos(x + b)$

$x \in [2, 8]$ $\Delta x = 1$ b – будь-яке число.

Задача 6. Знайти мінімум значення функції y та значення відповідного аргументу

y=ae^-sinα $a \in [m, k]$ $h_a = h$ *m, k, α, h* – будь-які , *m<k.*

Задача 7. Визначити, яким за порядком є мінімальне значення функції

y=ae^-b+ce^-d $b \in [f, g]$ $h_b = h$.

Задача 8. Визначити різницю між максимальним та мінімальним значеннями функції

$q=be^{-\sin\varphi}$ $b=2\sqrt{y}$ $\varphi\in[0,\,0{,}5]$ $h_\varphi=0{,}1$ $y\in[0,\,5]$ $h_y=0{,}5$.

Задача 9. Визначити значення аргументу, при якому функція досягає максимуму

$$y=-\frac{\sin\alpha}{\cos\beta}\quad \alpha\in[0,\,1]\quad h_\alpha=0{,}2.$$

Задача 10. Визначити максимальне значення функції та значення відповідного аргументу

$$Z=\frac{ae^x}{\sin ax}\quad x\in[m,\,n]\quad h_x=0{,}1m$$ m, n – будь-які a=0,2.

Задача 11. Обчислити $x=\left(\max+\frac{a+b}{\max}\right)$, де max – максимальне значення функції, max≠0 $Z=e^{-ay}$ $y\in[-5,\,6]$ $h_y=0{,}5$.

Задача 12. Знайти значення аргументу, при якому функція $q=(a+b)e^{-2y}$ досягає мінімального значення $y\in[-2,\,2]$ $h_y=0{,}01$.

Задача 13. Знайти різницю між максимальним та мінімальним значеннями функції

$y=|b|e^{-\alpha(x+d)}$ $\alpha\in[-c,\,c]$ $h_\alpha=0{,}5$.

Задача 14. Знайти значення аргументу, при якому функція $Z=\frac{|a|e^{-cx}}{a+b}$ досягає максимального значення $c\in[-5,\,0]$ $h_c=0{,}01$.

Задача 15. Знайти порядковий номер максимального значення та аргумент при мінімальному значенні функції

$T=a\cos b$ $b\in[b_1,\,b_2]$ $\Delta b=h$ a, b_1, b_2, h – будь-які числа.

Лабораторна робота 8

Завдання. Масив X містить N елементів. *Скласти схему алгоритму та програму введення значень елементів масиву та виконати обчислення згідно із умовами задач, наведених нижче.*

*Задача 1.*Визначення добутку елементів з ключовою ознакою

$12 \le X(i) \le 33$.

Задача 2. Визначення середнього арифметичного елементів масива з ключовою ознакою $-2 \le X(i) \le 13$.

Задача 3. Визначення різниці між добутками від'ємних та додатних елементів.

Задача 4. Визначення мінімального елемента.

Задача 5. Визначення кількості елементів, що більші середнього арифметичного.

Задача 6. Визначення різниці між максимальним та мінімальним елементами.

Задача 7. Визначення суми елементів з ключовою ознакою $X(i) > 56$ або $X(i) < -13$, розташованих на парних місцях.

Задача 8. Визначення добутку елементів з ключовою ознакою $1 \le X(i) \le 23$, розташованих на непарних місцях.

Задача 9. Визначення середнього арифметичного елементів масиву, розташованих на парних місцях.

Задача 10. Визначення максимального елемента масиву з непарним індексом.

Задача 11. Визначення кількості елементів з парними індексами, що менші середнього арифметичного усіх елементів.

Задача 12. Визначення суми елементів з ключовою ознакою –3,14 ≤ X(i) ≤ 6,28, розташованих на непарних місцях.

Задача 13. Формування масиву Z з елементів X(i), ключова ознака яких X(i) > 25 або X(i) < -3 або X(i) = 0.

Задача 14. Формування масиву Y з елементів X(i), ключова ознака яких 1<X(i) < 125, розташованих на парних місцях.

Задача 15. Формування масиву D з елементів X(i), ключова ознака яких X(i) <2 і X(i)>-20 або X(i) = 5.

Завдання. *Скласти схему алгоритму та програму.*

Задача 1. Задані масиви $\{x_i\},\{y_i\},i=1,10$ та довільні змінні a,b. Сформувати масив $\{c_i\},i=1,10$ та надрукувати його елементи

$$c_i=\frac{a\sin x_i+b\cos y_i}{x_i^2+1}.$$

Задача 2. Заданий масив $\{x_i\},i=1,n$, де n - довільна змінна. Сформувати масив $\{d_i\},i=1,n$ та надрукувати його елементи

$$d_i=\begin{cases}x_i^2, & якщо\ x_i<0\\ |x_i|, & якщо\ x_i\ge 0\end{cases}$$

Задача 3. Заданий масив $\{z_k\},k=1,15$. Сформувати масив $\{d_k\},k=1,15$ та надрукувати його елементи

$$d_k=\begin{cases}z_k+1, & якщо\ z_k<20\\ \ln z_k^5, & якщо\ z_k>20\\ z_k^2, & якщо\ z_k=20\end{cases}$$

Задача 4. Задані масиви $\{x_i\},\{a_i\},i=1,12$. Сформувати масив $\{f_i\},i=1,12$ та надрукувати його елементи

$$f_i=\ln x_i+2\sum_{j=1}^{8}a_j.$$

Задача 5. Задані масиви $\{a_i\},i=1,10,\{b_j\},j=1,12$. Сформувати масив $\{c_i\},i=1,10$ та надрукувати його елементи

$$c_i=\frac{a_i+\prod_{k=1}^{7}b_k}{\sum_{k=1}^{9}b_k}.$$

Задача 6. Заданий масив $\{t_i\},i=1,n$, де n - довільна змінна. Сформувати масив $\{r_i\},i=1,n$ та надрукувати його елементи

$$r_i = \begin{cases} t_i^{32}, & якщо\ t_i < 10 \\ |t+1_i|, & якщо\ t_i \geq 10 \end{cases}$$

Задача 7. Заданий масив $\{s_k\}, k = 1{,}18$. Сформувати масив $\{q_k\}, k = 1{,}18$ та надрукувати його елементи

$$q_k = \begin{cases} s_k^5 + 1, & якщо\ s_k < 100 \\ \lg s_k^5, & якщо\ s_k > 100 \\ s_k^2, & якщо\ s_k = 100 \end{cases}$$

Задача 8. Задані масиви $\{a_i\}, i = 1{,}n$, $\{b_j\}, j = 1{,}m$ n,m – довільні змінні. Сформувати масив $\{c_i\}, i = 1{,}m$ та надрукувати його елементи

$$c_i = \frac{b_i + \prod_{k=1}^{n} a_k}{\sum_{k=1}^{n} a_k}.$$

Задача 9. Заданий масив $\{z_k\}, k = 1{,}15$. Сформувати масив $\{d_k\}, k = 1{,}15$ та надрукувати його елементи

$$d_k = \begin{cases} z_k + \prod_{j=1}^{10} z_j, & якщо\ z_k < 20 \\ \ln z_k^5, & якщо\ z_k > 20 \\ z_k^2 - \sum_{j=3}^{8} z_j, & якщо\ z_k = 20 \end{cases}$$

Задача 10. Заданий масив $\{x_i\}, i = 1{,}n$, де n- довільна змінна. Сформувати масив $\{d_i\}, i = 1{,}n$ та надрукувати його елементи

$$d_i = \begin{cases} x_i^2 + \sum_{k=2}^{5} x_k, & якщо\ x_i < 0 \\ |x_i| - \prod_{j=1}^{4} x_j, & якщо\ x_i \geq 0 \end{cases}$$

Задача 11. Заданий масив $\{t_i\}, i = 1{,}n$, де n- довільна змінна. Сформувати масив $\{r_i\}, i = 1{,}n$ та надрукувати його елементи

$$r_i = \begin{cases} t_i^3, & якщо\ \sum_{k=1}^{6} t_k < 10 \\ |t+1_i|, & якщо\ \sum_{k=1}^{6} t_k \geq 10 \end{cases}$$

Задача 12. Задані масиви $\{x_i\}, \{y_i\}, i = 1{,}10$ та довільні змінні a,b. Сформувати масив $\{c_i\}, i = 1{,}10$ та надрукувати його елементи

$$c_i = \frac{a \sin \prod_{k=2}^{7} x_k + b \cos \sum_{n=1}^{5} y_n}{x_i^2 + 1}.$$

Задача 13. Заданий масив $\{s_k\}, k = 1{,}18$. Сформувати масив $\{q_k\}, k = 1{,}18$ та надрукувати його елементи

$$q_k = \begin{cases} s_k^5 + 1 + \sum_{i=1}^{5} s_i, & \textit{якщо } s_k < 200 \\ \lg s_k^5 + \sum_{[=2}^{7} s_i, & \textit{якщо } s_k > 200 \\ s_k^2 - \prod_{i=3}^{10} s_i, & \textit{якщо } s_k = 200 \end{cases}$$

Задача 14. Задані масиви $\{x_i\}, \{y_i\}, i = 1{,}20$. Сформувати масив $\{z_i\}, i = 1{,}20$ та надрукувати його елементи

$$z_i = \begin{cases} x_i - \prod_{j=1}^{5} y_j, & \textit{якщо } \sum_{k=1}^{5} x_k \le \prod_{j=1}^{5} y_j \\ \sum_{k=1}^{5} x_k + y_i, & \textit{якщо } \sum_{k=1}^{5} x_k > \prod_{j=1}^{5} y_j \end{cases}$$

Задача 15. Сформувати масив $\{x_j\}$ з від'ємних, масив $\{z_i\}$ з додатних і масив $\{a_n\}$ з нульових елементів масиву $\{y_k\}, k = 1{,}25$.

Лабораторна робота 9

Завдання. Заданий двовимірний масив числових значень з довільною кількістю рядків і стовпців. *Згідно з завданням скласти схему алгоритму та програму і виконати відповідну обробку даних у цьому масиві.*

Задача 1. Визначити кількість та суму від'ємних значень у кожному стовпці.

Задача 2. Знайти мінімальне значення на побічній діагоналі масиву.

Задача 3. Обчислити середнє арифметичне додатних елементів, розташованих під головною діагоналлю масиву.

Задача 4. Обчислити добуток додатних елементів, розташованих під головною діагоналлю масиву.

Задача 5. Визначити число елементів першого рядка, що за значенням менше, ніж середнє арифметичне в цьому ж рядку.

Задача 6. Знайти мінімальне і максимальне значення на побічній діагоналі масиву.

Задача 7. Визначити число елементів кожного рядка, що за значенням більше, ніж середнє арифметичне в цьому ж рядку.

Задача 8. Обчислити середнє арифметичне значення додатних елементів у кожному рядку.

Задача 9. Обчислити середнє арифметичне від'ємних елементів, розташованих над головною діагоналлю масиву.

Задача 10. Обчислити середнє арифметичне додатних елементів, розташованих над головною діагоналлю масиву.

Задача 11. Знайти різницю між мінімальним і максимальним значенням елементів на побічній діагоналі масиву.

Задача 12. Знайти різницю між мінімальним і максимальним значенням елементів масиву.

Задача 13. Підрахувати кількість додатних та від'ємних елементів масиву.

Задача 14. Знайти різницю між середнім арифметичним елементів у парних та непарних стовпцях.

Задача 15. Визначити суму від'ємних значень у кожному стовпці.

Лабораторна робота 10

Завдання. Задані списки (масиви) числових значень з довільною кількістю елементів. *Згідно з умовою задачі скласти схему алгоритму та програм , виконати над даними списками операції та вивести списки на друк до і після операцій.*

Задача 1. Вставити в список L1 за першим входженням заданого елемента список L2, який перед цим упорядкувати по убуванню.

Задача 2. "Перевернути" список L1 - тобто змінити порядок в ньому елементів на протилежний.

Задача 3. У списку L2 виконати упорядкування по зростанню.

Задача 4. Додати в кінець списку L1 всі елементи списку L2, упорядкувавши їх перед цим по убуванню.

Задача 5. Перевірити на нерівність списки L1 і L2, упорядкувавши їх перед цим по убуванню.

Задача 6. Список M1 упорядкувати по зростанню, список M2 упорядкувати по убуванню. Визначити, чи входить список M1 у M2 і навпаки.

Задача 7. Додати в початок списку L1 новий елемент, упорядкувавши L1 перед цим по убуванню.

Задача 8. У списку L2 виконати упорядкування по зростанню, а у списку L1 - по убуванню.

Задача 9. У списку L виконати упорядкування по зростанню та по убуванню.

Задача 10. Додати новий елемент в упорядкований по зростанню список M2 на відповідне місце.

Задача 11. Підрахувати кількість однакових елементів в упорядкованих по зростанню списках L1 та L2.

Задача 12. Додати в кінець списку L1 елементи списку L2, які більші заданого елемента. Отриманий список упорядкувати по убуванню.

Задача 13. Додати в початок списку L1 елементи списку L2, які більші заданого елемента. Отриманий список упорядкувати по зростанню.

Задача 14. Додати новий елемент в упорядкований по убуванню список L2, на відповідне місце.

Задача 15. Додати в початок списку L1 всі елементи списку L2. Отриманий список упорядкувати по зростанню.

Лабораторні роботи 11, 12

Завдання. *Скласти схему алгоритму та програму, згідно з завданням виконати обробку даних у масивах. Введення даних та виведення результатів здійснити в головній програмі.*

Задача 1. За допомогою процедури-підпрограми сформувати масив $\{x_j\}$, j=1,n з максимальних елементів кожного стовпця масиву $\{y_{ij}\}$, i=1,k; j=1,n.

Задача 2. За допомогою процедури-підпрограми транспонувати (поміняти місцями рядки і стовпці) та підрахувати суму всіх додатних елементів масиву $\{s_{ij}\}$, i=1,3; j=1,4.

Задача 3. За допомогою процедури-підпрограми сформувати масив $\{c_i\}$, i=1,n з середніх арифметичних кожного рядка масиву $\{d_{ij}\}$, i=1,n; j=1,m.

Задача 4. За допомогою процедури-підпрограми знайти мінімальне та максимальне значення масиву$\{a_{ij}\}$, i=1,3; j=1,4.

Задача 5. За допомогою процедури-функції знайти найбільше серед елементів парних стовпців масиву$\{z_{ij}\}$, i=1,3; j=1,4.

Задача 6. За допомогою процедури-функції знайти різницю між найбільшим і найменьшим значеннями масиву$\{b_{ij}\}$, i=1,m; j=1,n.

Задача 7. За допомогою процедури-функції знайти суму елементів побічної діагоналі масиву $\{x_{ij}\}$, i=1,n; j=1,n.

Задача 8. За допомогою процедури-функції знайти номер рядка, в якому міститься максимальний елемент масиву $\{y_{ij}\}$, i=1,3; j=1,4.

Задача 9. За допомогою процедури-функції знайти кількість ненульових елементів, розташованих у непарних стовпцях масиву $\{f_{ij}\}$, i=1,n; j=1,m.

Задача 10. За допомогою процедури-функції підрахувати кількість елементів парних рядків, масиву $\{c_{ij}\}$, i=l,n; j=1,k, які задовольняють умові $b<c_{ij}<d$.

Задача 11. За допомогою процедури - підпрограми сформувати з масивів $\{a_{ij}\}$ та $\{b_{ij}\}$ i=l,n; j=1,m масив $\{c_{ij}\}$, i=l,n; j=1,m , де $\{c_{ij}\}=\max(a_{ij}, b_{ij})$.

Задача 12. За допомогою процедури - підпрограми сформувати масив $\{ y_t \}$ з -елементів $2<x_{ij}<15$, масиву $\{x_{ij}\}$i=l,k; j=1,k+1 та підрахувати кількість елементів в $\{ y_t \}$.

*Задача 13. З*а допомогою процедури-функції підрахувати кількість рядків, в яких нема від'ємних елементів та кількість стовпців, в яких нема додатних елементів масиву $\{w_{ij}\}$, i=l,3; j=1,4.

Задача 14. За допомогою процедури-функції знайти добуток усіх елементів масиву $\{a_{ij}\}$, i=l,n; j=1,m, сума індексів яких дорівнює довільному числу b.

Задача 15. За допомогою процедури-підпрограми сформувати масив$\{z_{ij}\}$, i=l,4; j=1,3, де $z_{ij=}\begin{cases} x_{ij,} & \text{якщо } x_{ij}>y_{ij}, \\ y_{ij,} & \text{якщо } x_{ij}\le y_{ij}, \end{cases}$

використовуючи масиви $\{x_{ij}\}$ та $\{y_{ij}\}$, i=l,4; j=1,3.

Лабораторна робота 13

Завдання. *За допомогою оператора форматного виведення побудувати таблицю, що містить вихідні дані, наведені у задачі.*

Задача 1. Задано інформацію про студентів:
- Прізвище (20 символів)
- Ім'я (10)
- По батькові (10)
- Рік народження (4)
- Група (7)

*Задача 2.*Задано інформацію про слухачів курсів:
- Прізвище (15 символів)
- Ім'я (10)
- По батькові (10)
- Стать (Ч або Ж)
- Група (7)

Задача 3. Задано інформацію про студентів:
- Прізвище (15 символів)
- Ім'я (10)
- По батькові (10)
- Стать (Ч або Ж)
- Число годин пропусків (3)
- Факультет (4)
- Група (7)

Задача 4. Задано інформацію про студентів:
- Прізвище (15 символів)
- Ім'я (10)
- По батькові (10)
- Група (7)
- Оцінка з фізики (1)
- Оцінка з хімії (1)
- Оцінка з ОТ (1)

Задача 5. Задано інформацію про результати змагань з бігу:
- Прізвище (22 символи)
- Ім'я (12)
- По батькові (12)
- Команда (10)
- Результат (5)

Задача 6. Задано інформацію про результати змагань з кросу:
- Прізвище (15 символів)
- Ім'я (10)
- По батькові (10)
- Команда (10)
- Стать (1)
- Результат (5)

Задача 7. Задано інформацію про результати змагань з триборства: біг, плавання, конкур (очки):

Прізвище (15 символів)
Ім'я (10)
По батькові (10)
Результат бігу (5)
Результат плавання (5)
Результат конкуру (5)

Задача 8. Задано інформацію про продукти, що надійшли на склад:

Найменування (15 символів)
Артикул (10)
Дата продажу (ММ-ДД- РРРР - 10)
Обсяг (5)
Ціна (5)

Задача 9. Задано інформацію про студентів:

Прізвище (15 символів)
Ім'я (10)
Оцінка з фізики (1)
Оцінка з хімії (1)
Оцінка з ОТ (1)

Задача 10. Задано інформацію про ціни продуктів у магазині:

Найменування (15 символів)
Дата продажу (ММ-ДД-РРРР - 10)
Обсяг (5)
Ціна (5)

Задача 11. Задано інформацію про ціни запасів продуктів на складі:

Найменування (15 символів)
Дата продажу (ММ-ДД-РРРР - 10)
Обсяг (5)
Ціна (5)

Задача 12. Задано інформацію про результати змагань з шахів:

Прізвище (15 символів)
Країна (15)
Результат - очки (4)
Число перемог (2)
Число нічиїх (2)
Число поразок (2)

Задача 13. Задано інформацію про вартість запчастин рухомих одиниць на складі:

Найменування (15 символів)
Дата надходження (запчастин ММ-ДД- РРРР - 10)
Обсяг (число од. -5)
Ціна од.(5)

Задача 14. Задано інформацію про вартість букс:

Найменування (15 символів)
Дата надходження (ММ-ДД-ГГГГ - 10)
Обсяг (число од. -5)
Ціна од.(5)

Задача 15. Задано інформацію про результати змагань зі штанги: жим і ривок:

Прізвище (15 символів)
Ім'я (10)
По батькові (10)
Результат жиму (5)
Результат ривка (5)

Лабораторна робота 14

Завдання. *Скласти схему алгоритму та программу.*

Задача 1. Замінити в тексті цифру 1 на цифру 2, підрахувати, скільки зроблено замін і вставити після перших десяти символів 4 пробіли.

Задача 2. Підрахувати, у скільки разів буква "К" зустрічається в тексті частіше, ніж буква "Ф", замінити великі букви на маленькі.

Задача 3. Розділити текст на дві рівних половини, вставити між ними 6 пробілів; в 1-й половині замінити великі букви на маленькі, а в 2-й - маленькі на.великі.

Задача 4. Визначити, у скільки разів одне задане слово зустрічається в тексті частіше, ніж інше задане слово. На початку тексту вставити ланцюжок символів "<<<", а наприкінці - ">>>".

Задача 5. Задано два тексти. Визначити, у якому з них більше речень. Речення закінчується крапкою.

Задача 6. Визначити, скільки разів у тексті зустрічається буквосполучення "ма"; замінити великі букви на маленькі і перші п'ять символів відокремити від іншого тексту символами ***.

Задача 7. Визначити, скільки відсотків від загального числа слів становить слово 'вухо'. У тексті між словами один пробіл.

Задача 8. Задано два тексти. Визначити, у скільки разів буква "м" зустрічається в 1-му тексті частіше, ніж в 2-му. Об'єднати обидва тексти в один і замінити в отриманому тексті великі букви на маленькі.

Задача 9. Видалити з тексту всі пробіли. Підрахувати число символів до і після видалення (початкові й кінцеві пробіли не враховувати).

Задача 10. Визначити різницю між кількістю букв "А" і "Е" у тексті. Замінити в перших п'яти символах тексту великі букви на маленькі, а замість останніх десяти символів вставити пробіли.

Задача 11. Підрахувати кількість речень у тексті і число слів у 1-му реченні.

Задача 12. Підрахувати число символів у 1-му реченні тексту і число слів в іншій частині тексту.

Задача 13. Розділити текст на половини і підрахувати, скільки символів, що не є пробілами, містить 1-а половина тексту й скільки пробілів- 2-а половина.

Задача 14. Задано два тексти. Підрахувати, на скільки слів у 1-му тексті більше (або менше) чим в 2-му. Об'єднати обидва тексти в один, вставивши між ними ланцюжок символів "&&&".

Задача 15. Визначити, на скільки букв "А" у тексті більше, ніж букв "Я". Замінити великі букви "М" на маленькі.

Лабораторна робота 15

Завдання. *Користуючись умовою завдання відповідного варіанту лабораторної роботи 13 скласти схему алгоритму та програму.*

Задача 1

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список студентів, яким немає 18 років.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список студентів з парними порядковими номерами.

Задача 2

1. Створити файл послідовного доступу, що містить перераховану інформацію. Прочитати з файла й роздрукувати список жінок.

2. Створити файл прямого доступу, що містить перераховану інформацію. Прочитати з файла дані й перевести всіх жінок в іншу групу. Зберегти список жінок у новому файлі й роздрукувати цей список.

Задача 3

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список студентів заданої групи.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список студентів, число годин пропусків у яких більше 30.

Задача 4

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список продуктів, що надійшли сьогодні.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані із заданим найменуванням.

Задача 5

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список студентів, які добре навчаються.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список студентів, які мають заборгованості.

Задача 6

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список спортсменів, чий результат краще 11 с.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список спортсменів з команди "Динамо".

Задача 7

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати окремо список чоловіків і жінок.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані спортсмена з найкращим результатом.

Задача 8

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список спортсменів з кращими результатами з кожного виду спорту окремо .

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані спортсмена з найгіршим результатом.

Задача 9

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список студентів, які:

а) найкраще навчаються;

б) добре начаються.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список студентів, прізвище яких починається на букву 'А' .

Задача 10

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла збережені дані й роздрукувати суму виторгу за поточний день.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані із заданим найменуванням і ціною не вище заданої.

Задача 11

1. Створити файл послідовного доступу, що містить вихідні дані. Ціна збільшується на 1грн. у день. Прочитати з файла збережені дані, скоректувати їх (5 днів), знову записати на місце й роздрукувати скоректовані записи. Визначити загальну вартість продуктів через 5 днів.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані про продукти з найвищою ціною.

Задача 12

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані про найкращого й найгіршого спортсменів (припускаємо відсутність однакових результатів).

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані спортсменів заданої країни.

Задача 13

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла збережені дані, скорегувати їх з урахуванням амортизації (зменшити ціну на 5 грн), знову записати на місце й роздрукувати скореговані записи. Визначити загальну вартість запчастин.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані по заданому найменуванню.

Задача 14

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла збережені дані. Визначити вартість заданого найменування й загальну вартість.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані по буксах, що надійшли сьогодні.

Задача 15

1. Створити файл послідовного доступу, що містить вихідні дані. Прочитати з файла й роздрукувати список спортсменів із найкращими результатами по кожному виді окремо.

2. Створити файл прямого доступу, що містить вихідні дані. Прочитати з файла й роздрукувати дані спортсмена з найкращим результатом.

Лабораторна робота 16

1. Зобразити точку на екрані з координатами (X,Y), де X=Y=номер за списком*5 при різних режимах екрана (1,2,12).

2. Зобразити точку на екрані з координатами, зміщеними на X-3 і Y+5 одиниць, задаючи їх:

- в абсолютній формі;
- у відносній формі

3. Нарисувати відрізок, що з'єднує дві нарисовані в п.п. 1,2 точки, операторами:

- LINE
- DRAW

4. Із точки з координатами (X,Y), де X=Y=номер за списком*7 (лівий верхній кут) нарисувати квадрат зі стороною X*2 операторами:

- LINE
- DRAW

5. Нарисувати першу букву свого імені оператором DRAW, повернути її на 90 градусів і розфарбувати будь-яким кольором.

6. Нарисувати окружність оператором CIRCLE з радіусом R=номер за списком.

7. Реалізувати рух точки від (X,Y) до (3*X, 3*Y)

8. нарисувати:

1) ромашку;
2) робота;
3) олімпійські кільця;
4) сходи;
5) будиночок;
6) 5-кінцеву зірку;
7) український прапор;
8) куб;
9) повітряного змія;
10) стрілковий годинник.

Кожний пункт оформити у вигляді програми з відповідним номером і ім'ям, відлагодити і продемонструвати працюючу програму.

Комплексне завдання

Заданий двовимірний масив ***X***, що складається з ***N*** рядків і ***M*** стовпців (***N*** , ***M***>=5).

1) ЗАПОВНИТИ масив ***X*** цілими числами з діапазону від ***a1*** до ***a2***;

2) ЗАМІНИТИ дані рядка №/стовпця № їхніми порядковими номерами в рядку/стовпці;

3) ЗНАЙТИ різницю між найбільшим/найменшим елементом, що лежить на головній діагоналі і середнім арифметичним додатних/від'ємних елементів, розташованих у парних/непарних рядках/стовпцях;

4) СФОРМУВАТИ одновимірний масив ***B*** з елементів масиву ***X***, що належать діапазону ***a4*** або значенням ***a5***, використовуючи процедуру-підпрограму;

5) ЗНАЙТИ в сформованому масиві ***B*** найбільший/найменший елемент і його порядковий номер;

6) ВІДСОРТУВАТИ по по спаданню/зростанню елементи масиву ***B***, розташовані вище/нижче найбільшого елемента;

7) ПОБУДУВАТИ в графічному режимі графік залежності значення елемента від його номера рядка/стовпця і побудувати лінію середнього значення елемента для даного рядка/стовпця;

8) СФОРМУВАТИ ФАЙЛ послідовного доступу з елементів масиву ***X***, що належать діапазону ***a4***; вивести на екран зі сформованого файла всі парні/непарні числа;

9) ПЕРЕВІРИТИ роботу програми для масиву X, заповненого випадковими дійсними числами, з діапазону від ***a1*** до ***a2***.

4.2. СПИСОК ТЕСТОВИХ ПИТАНЬ

1. Проект в Visual Basic - це:

1. **один або декілька програмних модулів;**
2. **одна або кілька екранних форм;**
3. **сукупність частин, що становлять Windows-додаток;**
4. **інтерфейс програми + програмний код.**

2. Visual Basic зберігає кожний проект у файлі з розширенням:

1. **.VB6p;**
2. **.frm;**
3. **.bas;**
4. **.frx.**

3. Програмний код проекта:

1. **існує сам по собі;**
2. **прив'язаний до окремих об'єктів і не відірваний від форми;**
3. **прив'язаний до окремих об'єктів і не пов'язаний з формою;**
4. **прив'язаний тільки до форми.**

4. Програмний модуль Visual Basic має розширення:

1. **.VB6p;**
2. **.frm;**
3. **.bas;**
4. **.frx.**

5. На якому етапі розроблення програми в середовищі Visual Basic можна обрати метод об'єкта:

1. **на етапі проектування інтерфейсу;**
2. **на етапі кодування тексту програми.**

6. На якому етапі розроблення програми в середовищі Visual Basic можна змінити властивості об'єкта:

1. **тільки на етапі проектування інтерфейсу;**
2. **тільки на етапі кодування тексту програми;**
3. **і на етапі проектування інтерфейсу, і на етапі кодування тексту програми.**

7. З яких двох основних частин складається інтерфейс користувача:

1) **з простого інтерфейсу й посібника з використання цього інтерфейсу;**
2) **зі складного меню і кнопки "Вихід із програми";**
3) **з монітора і клавіатури;**
4) **з форм і об'єктів.**

8. Як можна змінити властивості об'єктів:

1. **ніяк, тому що вони встановлюються раз і назавжди;**
2. **тільки використовуючи вікно Properties під час створення інтерфейсу;**
3. **тільки за допомогою програмних кодів у процесі виконання програми;**
4. **і в процесі створення інтерфейсу, і в процесі виконання програми.**

9. Навіщо потрібні процедури обробки подій:

1) вони переривають роботу програми;
2) вони повідомляють комп'ютеру, як потрібно реагувати на події, які ініціюються користувачем за допомогою об'єктів інтерфейсу;
3) вони повідомляють про помилки користувача;
4) вони повідомляють про несправності в роботі комп'ютера.

10. Що таке форма:

1) філософське поняття, протилежне "змісту" програми;
2) об'єкт, що з'являється на екрані при запуску Visua1 Basic і призначений для введення даних;
3) термін Visua1 Basic, що позначає вікно програми;
4) об'єкт, що з'являється на екрані під час запуску програми і призначений для виводу результатів роботи програми.

11. Елемент керування - це:

1) об'єкт, за допомогою якого програма запускається на виконання;
2) об'єкт, що є елементом графічного інтерфейсу додатка й реагуючий на події;
3) термін Visua1 Basic, що позначається значком на панелі інструментів Toolbox;
4) 4 об'єкт, що з'являється на екрані при запуску програми й призначений для зупинки програми.

12. Для яких цілей можуть бути використані текстові поля:

1) ними можна заповнити вікно форми;
2) для виведення інформації на екран і для одержання даних від користувача;
3) тільки для виведення інформації на екран;
4) тільки для одержання даних від користувача;
5) краще їх не використовувати.

13. Чим корисні меню, що розкриваються:

1) тим, що в них завжди можна знайти команду He1p і Exit;
2) з їхньою допомогою можна розбити на групи й компактно розмістити команди, що виконують подібні дії;
3) нічим;
4) створюють у користувача ілюзію того, що дана програма дуже серйозна;
5) гарантують правильність роботи з програмою.

14. Що зберігається у файлі форми (з розширенням. frm):

1) форма;
2) програмний код, пов'язаний з формою;
3) дані про об'єкти, які розташовані на формі;
4) дані про форму, об'єкти, розташовані на ній, і програмний код, пов'язаний з формою.

15. Що таке подія в Visual Basic:

1) **одержання даних;**
2) **будь-які зміни, які ініціюються користувачем або системою, на які може реагувати програма;**
3) **вихід з ладу монітора;**
4) **при роботі програми не відбувається подій.**

16. Встановіть відповідність між подією й тим, коли вона відбувається:

Подія	Відбувається
LOAD	при зміні значення властивості Visible керуючого елемента або документа на True
CLICK	коли об'єкт стає поточним вікном
MOUSEDOWN	при натисканні кнопки миші в момент, коли покажчик миші знаходиться на керуючому елементі
CHANGE	коли об'єкт перестає бути поточним вікном
DBLCLICK	при натисканні кнопки миші
SHOW	при переміщенні повзунка на смузі прокручування або елементі, що містить смугу прокручування
SCROLL	при завантаженні форми
	при зміні вмісту керуючого елемента
	при подвійному натисканні кнопки миші в момент, коли покажчик миші знаходиться на керуючому елементі

17. Встановіть відповідність між методом і дією:

Метод	Дія
CLS	переміщує форми, MDI-форми й керуючі елементи
PRINT	видаляє сліди текстового й графічного виводу з робочої поверхні форми або керуючого елемента Picture
REFRESH	завантажує рисунок з файла й виводить його на робочу поверхню форми, картинки або об'єкта Printer
SHOW	примусово перерисовує форму або керуючий елемент відповідно до його поточного вмісту
SIZE	відображає на екрані форму або MDI-форму
	змінює розміри обумовленого користувачем керуючого елемента
	здійснює друк на формі або вікні PictureBox

18. Що з перерахованого є назвами властивостей об’єктів:

1) **TEXT;**
2) **INTEGER;**
3) **DATE;**
4) **NAME;**
5) **CAPTION;**
6) **STRING;**
7) **SINGLE.**

19. Маємо фрагмент програми:

```
Dim int As Integer
Dim int As Integer
Dim int As Integer
Private Sub Commandl_Click()
    int = 2
    int = 3
    int = int/int
    Form1. Print int
End Sub
```

Ім'ям процедури-події в програмі є:

1) **Forml;**
2) **Print;**
3) **Commandl_Click();**
4) **int.**

20. У властивості Name об'єкта прийнято використовувати префікс:

Об'єкт	Префікс
Форма	fra
Мітка	dir
Текстове поле	txt
Вікно списку	cbo
Список дискових накопичувачів	1st
	drv
	frm
	lbl

21. Значення якого з перерахованих властивостей указується у програмі при звертанні до об'єкта:

1) **CAPTION;**
2) **ALIGNMENT;**
3) **TEXT;**
4) **NAME;**
5) **ENABLED.**

22. Значення якої властивості виводиться в рядок заголовка форми:

1) **CAPTION;**
2) **ALIGNMENT;**
3) **TEXT;**
4) **NAME;**
5) **MULTILINE.**

23. Значення якої властивості визначає варіант вирівнювання тексту в мітці:

1) **CAPTION;**
2) **ALIGNMENT;**
3) **TEXT;**
4) **NAME;**
5) **MULTILINE.**

24. Значення якої властивості дозволяє (або забороняє) виведення декількох рядків у текстовому вікні:

1) **CAPTION;**
2) **ALIGNMENT;**
3) **TEXT;**
4) **NAME;**
5) **MULTILINE.**

25. Виберіть правильні імена змінних:

1) **Print;**
2) **Numer;**
3) **One.Year;**
4) **Yearl;**
5) **Процентна ставка;**
6) **Your_Name.**

26. Щоі з перерахованого є типоми даних:

1) **TEXT;**
2) **INTEGER;**
3) **DATE;**
4) **NAME;**
5) **CAPTION;**
6) **STRING;**
7) **SINGLE.**

27. Який тип даних використовується для оголошення символьних рядків:

1) **INTEGER;**
2) **BOOLEAN;**
3) **SINGLE;**
4) **STRING;**
5) **CURRENCY;**
6) **DATE.**

28. Який тип даних використовується для оголошення чисел у грошовому форматі:

1) **INTEGER;**
2) **BOOLEAN;**
3) **SINGLE;**
4) **STRING;**
5) **CURRENCY**

29. Який тип даних використовується для оголошення цілих чисел:

1) **INTEGER;**
2) **BOOLEAN;**
3) **SINGLE;**
4) **STRING;**
5) **CURRENCY;**
6) **DATE.**

30. Числа в мові Visual Basic розрізнюють як:

1) **натуральні й цілі;**
2) **цілі й десяткові;**
3) **натуральні й десяткові;**
4) **цілі й ірраціональні;**
5) **цілі й раціональні.**

31. Як ім'я змінної в мові VB6 не можна використовувати сполучення:

1) **AR;**
2) **BR;**
3) **OR;**
4) **PI;**
5) **WR.**

32. Що робить такий код:

ЩОЦЕ=VSCRBA.VALUE

1) **1 змушує Visual Basic вивести на екран підказку, що пояснює, чим є VSCRBAR.VALUE;**
2) **2 нічого;**
3) **змінної ЩОЦЕ привласнюється значення властивості VALUE об'єкта VSCRBAR;**
4) **якщо Visual Basic не може зрозуміти, для чого ви набралиVSCRBAR.VALUE, на екрані з'являється таке питання.**

33. Як дати зрозуміти програмі, що вводиться інформація, яка повинна бути сприйнята як рядок:

1) **наприкінці рядка поставити тире;**
2) **набрати текст без граматичних помилок;**
3) **текстова інформація повинна бути розміщена в лапках;**
4) **будь-яка інформація сприймається як рядок.**

34. Чи може в процесі виконання програми змінитися ім'я змінної:

1) **так;**
2) **ні.**

35. Що змінює операція присвоювання:

1) **значення змінної;**
2) **ім'я змінної ;**
3) **тип змінної;**
4) **нічого не змінює;**
5) **значення й тип одночасно.**

36. Чи може в процесі виконання програми змінитися значення змінної:

1) **так;**
2) **ні.**

37 . Запис "2000" у мові програмування VB6 являє собою:

1) **текстову константу з набору символів "2000";**
2) **ім'я змінної;**
3) **текстову константу з набору символів 2000;**
4) **число;**
5) **рік.**

38. Маємо фрагмент програми:

A$="5"
B$="4"
S$=A$+ B$
Print S$
Яке значення змінної S буде надруковане після його виконання:

1) **9;**
2) **54;**
3) **5;**
4) **4.**

39. Вкажіть послідовність команд, у результаті виконання яких значення змінних X і Y поміняються місцями:

1) **Y=X:B=X:X=Y;**
2) **B = X : X = Y : Y = X;**
3) **X = Y: Y = X;**
4) **C = X: X = Y: Y = C;**
5) **X = X + Y: Y= X -Y: X = X – Y**

40. Фрагмент програми

S = A: A = B: B = S
виконує:

1) **обмін значень змінних A, B;**
2) **присвоювання змінним A, B значення S;**
3) **заміну значення змінної A значенням змінної B;**
4) **у фрагменті не виконується ніяких дій;**
5) **заміну значення змінної B значенням змінної A.**

41. Що відбудеться в результаті виконання команди

PRINT "3*3="; 3*3:

1) **на екрані буде надруковане 9;**
2) **на формі буде виведене 9;**
3) **на екрані буде надруковано 3*3=9;**
4) **на формі буде виведене 3*3=9 ;**
5) **на екрані буде надруковано 3*3=3*3;**
6) **на формі буде виведено 3*3=3*3.**

42. У мові Visual Basic оператори, написані в одному рядку, розділяються:

1) **двокрапкою;**
2) **крапкою з комою;**
3) **комою;**
4) **пробілом;**
5) **круглими дужками.**

43. Для переносу довгих рядків програмного коду використовується:

1) **символ:**
2) **тире;**
3) **підкреслення;**
4) **кома;**
5) **пробіл;**
6) **&.**

44. Для чого коди програми потрібно супроводжувати коментарями:

1) **щоб дати короткий опис дій, які виконуються командами;**
2) **з їхньою допомогою можна пояснити користувачам програми для чого вона призначена;**
3) **цього не слід робити;**
4) **це просто традиція.**

45. Функція RND() повертає:

1) **знак числа;**
2) **найбільше ціле число, що не перевершує аргумент;**
3) **випадкове ціле число [1,100];**
4) **випадкове число в діапазоні [0,1];**
5) **модуль аргументу.**

46. Квадратний корінь аргументу повертає функція:

1) **EXP(X);**
2) **SQR(X);**
3) **INT(X);**
4) **FIX(X);**
5) **SGN(X)**

47. Найбільше ціле число, що не перевершує аргумент повертає функція:

1) **INT(X);**
2) **FIX(X);**
3) **CINT(X);**
4) **SGN(X);**
5) **ABS(X).**

48. Округлене число (з відкинутою дробовою частиною аргументу) повертає функція:

1) **INT(X);**
2) **FIX(X);**
3) **CINT(X);**
4) **SGN(X);**
5) **ABS(X).**

49. Аргумент, округлений до цілого за правилами округлення математики, повертає функція:

1) **INT(X);**
2) **FIX(X);**
3) **CINT(X);**
4) **SGN(X);**
5) **ABS(X).**

50. Укажіть синтаксично неправильний запис:

1) **X = Y * SIN(X)^ 2 + 4;**
2) **X = Y * SIN ^ 2 (X) + 4;**
3) **X = Y* SIN(X ^ 2) + 4;**
4) **X = Y^ 2 * SIN(X) + 4;**
5) **X = Y^ (2 * SIN(X) + 4).**

51. Яка з перерахованих операцій не є логічною:

1) **AND;**
2) **MOD;**
3) **NOT;**
4) **OR.**

52. Функція, що перетворює рядок символів у числове значення:

1) **ABS;**
2) **INT;**
3) **FIX;**
4) **RND;**
5) **STR;**
6) **VAL.**

53. Маємо фрагмент програми:

```
A=2
B=3
If A<B Then S=A*B Else S=A+B
Print S
```

Яке значення змінної S буде надруковане після виконання даного фрагмента програми:

1) **2;**
2) **3;**
3) **5;**
4) **6.**

54. Маємо фрагмент програми:

```
A=3
B=3
IF A<B THEN S=A*B ELSE S=A+B
Print S
```

Яке значення змінної S буде надруковане після виконання даного фрагмента:

1) **2;**
2) **3;**
3) **5;**
4) **6.**

55. Вкажіть послідовність команд, у результаті виконання яких буде знайдене найбільше значення змінних X і Y.

1) **MAX = Abs(X - Y) / 2 + (X + Y) / 2: Print MAX;**
2) **If > Y Then MAX = X : Print MAX;**
3) **If < Y Then MAX = Y: Print MAX;**
4) **If > YThen MAX = X: Print MAX Else MAX = Y: Print MAX,**
5) **MAX = ABS(X - Y) / 2 - (X + Y) / 2 : PRINT MAX.**

56. Умовними операторами є:

1) **IF ... THEN ... ELSE;**
2) **FOR ... NEXT;**
3) **WHILE ... WEND;**
4) **SELECT CASE ;**
5) **DO UNTIL ... LOOP.**

57. Умовний вираз Not ((X <= 10) Or (X >= 2)) може використовуватися для перевірки:

1) **чи належить число інтервалу (10; 20);**
2) **чи виходить число за межі інтервалу (10; 20);**
3) **чи належить число інтервалу [10; 20];**
4) **чи виходить число за межі інтервалу [10; 20].**

58. Умовний вираз "ABBA" > "ABA" має значення:

1) **TRUE;**
2) **FALSE.**

59. Маємо фрагмент програми:

```
A = 10
SELECT CASE A
CASE 1,10
 A=A*2
CASE IS > 10
A=A*2
 CASE ELSE
A=A*2
 END SELECT
PRINT A
```

Що буде надруковано на формі після виконання даного фрагмента:

1) **10;**
2) **20;**
3) **40;**
4) **80.**

60. Маємо фрагмент програми:

```
X=5:Y=10
IF X > 2 THEN
IF Y> 10 THEN
X=X*2
END IF
ELSE
X=X*5
END IF
```

Яке значення буде мати змінна X після виконання даного фрагмента:

1) **5;**
2) **10;**
3) **25;**
4) **125**

61. Маємо фрагмент програми:

```
X=9
IF X< 10 THEN
Y=1
ELSEIF X < 100 THEN
Y=2
ELSE
Y=3
END IF
```

Яке значення буде мати змінна Y після виконання даного фрагмента:

1) **1;**
2) **2;**
3) **3;**
4) **9.**

62. Що з перерахованого нижче є назвами операцій:

1) **AND;**
2) **RND;**
3) **NOT;**
4) **MOD;**
5) **STR;**
6) **ABS;**
7) **OR.**

63. Маємо фрагмент програми:

```
S=1
FOR N = 1 TO 3
S=S*N
 NEXT N
 PRINT S
```

Яке значення змінної S буде надруковане після виконання даного фрагмента:

1) **2;**
2) **3;**
3) **4;**
4) **6.**

64. Маємо фрагмент програми:

```
S=l
FOR N = 1 TO 3
 S=S+ N
NEXT N
PRINT S
```

Яке значення змінної S буде надруковане після виконання даного фрагмента:

1) **5;**
2) **7;**
3) **9;**
4) **12.**

65. Маємо фрагмент програми:

```
DIM K AS INTEGER, N AS INTEGER
DIM X AS INTEGER, Y AS INTEGER
 N=3
X=0
FOR K = 1 TO N
 Y=K*K
 Y=Y*ДО
 X=X+Y
NEXT K
PRINT "X="; X
```

Що буде надруковано в результаті виконання даного фрагмента:

1) **X = 27;**
2) **X = 9;**
3) **X = 36;**
4) **X = 3.**

66. Маємо фрагмент програми:

```
A=1
 S=O
DO WHILE S < 10
 S=S+A
A=A+A^2
LOOP
```

Скільки разів будуть виконані оператори тіла циклу при виконанні даного фрагмента:

1) **10;**
2) **3;**
3) **4;**
4) **5;**
5) **1.**

67. Маємо фрагмент програми:

```
N=3
S=0
FOR K = 1 TO N
S=S+K^2
NEXT
PRINT S
```

Яке число буде виведене при виконанні даного фрагмента:

1) **9;**
2) **27;**
3) **14 ;**
4) **6;**
5) **10**

68. В операторі циклу в мові програмування VB6 після службового слова STEP указується:

1) **вираз, що визначає початкове значення параметра циклу;**
2) **арифметичний вираз, значення якого визначає величину збільшення параметра циклу;**
3) **вираз, що визначає кінцеве значення параметра циклу;**
4) **логічний вираз, значення якого визначає величину збільшення параметра циклу.**

69. Маємо фрагмент програми:

```
N=5
 P=1
FOR K = 1 TO N
 P=P*K
 NEXT
PRINT P
```

Яке число буде надруковане в результаті виконання даного фрагмента:

1) **120;**
2) **15;**
3) **5;**
4) **20.**

70. Маємо фрагмент програми:

```
FOR K = 12 TO 1 STEP-3
 PRINT
NEXT K
```

Що з'явиться на екрані після виконання даного фрагмента:

1) **12, 9, 6, 3;**
2) **повідомлення про помилку;**
3) **12,8,4;**
4) **нічого.**

71. Маємо фрагмент програми:

```
I=0
DO UNTIL I > 5
I=I+1
 PRINT I
 LOOP
```

Що буде надруковано після виконання даного фрагмента:

1) **стовпчик чисел від 1 до 5;**
2) **стовпчик чисел від 1 до 6;**
3) **0;**
4) **1.**

72. Маємо фрагмент програми:

```
I = 0
DO UNTIL I > 5
 PRINT I
 LOOP
```

Що буде надруковано після виконання даного фрагмента:

1) **стовпчик чисел від 1 до 5;**
2) **стовпчик чисел від 1 до 6;**
3) **нескінченний цикл (друкується 0).**

73 . Маємо фрагмент програми:

```
DIM K AS INTEGER, M AS INTEGER
DIM N AS INTEGER
K=0
FOR M=1 TO 10
N=M MOD 3
IF N<>M MOD 5 THEN
        K=K+1
END IF
 NEXT M
PRINT K
```

Обчислене в програмі значення змінної K **дорівнює:**

1) **1;**
2) **9;**
3) **8;**
4) **6;**
5) **4.**

74. Маємо фрагмент програми:

```
DIM K AS INTEGER, M AS INTEGER
 DIM N AS INTEGER
K=0
FOR M=1 TO 10
        N=M MOD 4
IF N>=M MOD 3 THEN
K=K+l
END IF
NEXT M
PRINT K
```

Обчислене в програмі значення змінної К дорівнює:

1) **5;**
2) **7;**
3) **8;**
4) **6;**
5) **3.**

75. Маємо фрагмент програми:

```
DIM K AS INTEGER, M AS INTEGER
DIM N AS INTEGER
K=0
FOR M=1 TO 10
 N=INT(M/4)
 IF N=(11-M) MOD 2 THEN
 K=K+l
END IF
NEXT M
PRINT K
```

Обчислене в програмі значення змінної К дорівнює:

1) **5;**
2) **7;**
3) **8;**
4) **6;**
5) **4.**

76. Маємо фрагмент програми:

```
DIM K AS INTEGER, M AS INTEGER
DIM N AS INTEGER
K=0
FOR M=1 TO 10
N=M MOD 3
IF N>(11-M) MOD 2 THEN
K=K+l
END IF
NEXT M
PRINT K
```

Обчислене в програмі значення змінної К дорівнює:

1) **5;**
2) **7;**
3) **8;**
4) **6;**
5) **3.**

77. Маємо фрагмент програми:

```
DIM K AS INTEGER, M AS INTEGER
DIM N AS INTEGER
K=0
FOR M=1 TO 10
 N=INT(M/5)
IF N=M MOD 3 THEN
```

```
K=K+l
END IF
NEXT M
PRINT K
```

Обчислене в програмі значення змінної К дорівнює:

1) **5;**
2) **7;**
3) **2;**
4) **6;**
5) **3.**

78. Маємо фрагмент програми:

```
DIM K AS INTEGER, M AS INTEGER
DIM N AS INTEGER
K=0
FOR M=1 TO 10
 N=M MOD 3
IF N>(11-M) MOD 2 THEN
K=K+l
END IF
NEXT M
PRINT K
```

Обчислене в програмі значення змінної К дорівнює:

1) **5;**
2) **7;**
3) **8;**
4) **6 ;**
5) **3.**

79. Маємо фрагмент програми:

```
DIM K AS INTEGER, M AS INTEGER
DIM N AS INTEGER
K=0
FOR M=1 TO 10
 N=INT(M/5)
IF N=M MOD 3 THEN
K=K+l
END IF
NEXT M
PRINT K
```

Обчислене в програмі значення змінної К дорівнює:

1) **5;**
2) **7;**
3) **2;**
4) **6;**
5) **3.**

80. Керуючий елемент PictureBox призначений:

1) **для подання на робочій поверхні об'єкта геометричних фігур;**
2) **для відображення й модифікування тексту;**

3) **для відображення на екрані рисунка або значка;**
4) **для відображення на екрані рисунків, завантажених із графічних файлів різних форматів, і дозволяє рисувати на своїй поверхні за допомогою графічних методів;**
5) **для відображення лінії.**

81. Керуючий елемент Shape призначений:

1) **для подання на робочій поверхні об'єкта геометричних фігур;**
2) **для відображення й модифікування тексту;**
3) **для відображення на екрані рисунка або значка;**
4) **для відображення на екрані рисунків, завантажених із графічних файлів різних форматів, і дозволяє рисувати на своїй поверхні за допомогою графічних методів;**
5) **для відображення лінії.**

82. Керуючий елемент Image призначений:

1) **для подання на робочій поверхні об'єкта геометричних фігур;**
2) **для відображення й модифікування тексту;**
3) **для відображення на екрані рисунка або значка;**
4) **для відображення на екрані рисунків, завантажених із графічних файлів різних форматів, і дозволяє рисувати на своїй поверхні за допомогою графічних методів;**
5) **для відображення лінії .**

83. Чи можуть кілька кнопок на одній формі мати однакові значення властивості Name:

1) **ні;**
2) **так;**
3) **так, при різних значеннях властивості Index.**

84. Масив - це:

1) **пойменований набір однотипних даних;**
2) **обмежена апострофами послідовність будь-яких символів;**
3) **сукупність різнорідних даних, які описуються та оброблюються як єдине ціле;**
4) **набір однотипних файлів на диску;**
5) **набір змінних, що починаються з однієї букви.**

85. Маємо фрагмент програми:

```
M = A( 1)
FOR К=2 TO 8
IF M < A(К) THEN
 M = A(К)
END IF
NEXT
```

Скільки разів у програмі буде виконаний оператор M = A(К) при заданому масиві 3, 8, 7, 9, 4, 10, 2, 12:

1) **7;**
2) **8;**
3) **4;**
4) **1;**
5) **3.**

86 . Маємо фрагмент програми:

```
DIM A(10) AS INTEGER
DIM I AS INTEGER, К AS INTEGER
 DIM N AS INTEGER
К=0
FOR I=1 TO 10
A(I)=I+3
 NEXT I
FOR I= 1 TO 10
 N=A(I) MOD 4
IF N>(11-I) MOD 3 THEN
 К=К+l
END IF
NEXT I
PRINT К
```

Обчислене в програмі значення змінної К дорівнює:

1) **4;**
2) **3;**
3) **1;**
4) **2;**
5) **6.**

87. Маємо фрагмент програми:

```
DIM I AS INTEGER, S AS INTEGER
DIM X AS INTEGER
DIM A(10) AS INTEGER
FOR I = 1 TO 10
A(I) = 11 - 2 * 1
 NEXT I
S=0
FOR I = 1 TO 10
IF A(I) < 0 THEN
X = A(I) + A(I - 1)
ELSE
X = A( 11 - I) - A(I)
 END IF
S=S+X
NEXT I
PRINT "S=";S
```

Що буде надруковано в результаті виконання програми:

1) **S=50;**
2) **S= -50;**
3) **S=100;**
4) **S=0.**

88. Маємо фрагмент програми:

```
FOR N=1 TO 5
FOR K=1 TO 5
IF N>K THEN A(N, K)=(N + K)
ELSE
```

```
A(N, K)=(N*K)
 END IF
NEXT K
NEXT N
```

Яке значення буде мати елемент.масиву A(3,3) після виконання фрагмента програ.ми:

1) **9;**
2) **3;**
3) **6;**
4) **12.**

89. Маємо фрагмент програми:

```
FOR N=1 TO 5
FOR К=1 TO 5
IF N<K THEN A(N, K)=(N + К)
 ELSE
A(N, K)=(N*K)
 END IF
NEXT К
NEXT N
```

Яке значення набуде елемент масиву A(3,3) після виконаняя фрагмента програми:

1) **3;**
2) **6;**
3) **9;**
4) **12.**

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Браун С. Visual Basic 6: Учебный курс. – С.Пб.: Питер, 2006. – 574 с.

2. Волченков Н.Г. Программирование на Visual Basic 6. – В 3-х ч. – М.: ИНФРА-М, 2000. – 280 с.

3. Гончаров В.О., Меркулов В.С. Журнал № 4 до лабораторних робіт з дисципліни «Обчислювальна техніка та програмування» (вирішення задач в середовищі QBasic) для студентів усіх факультетів та форм навчання. – Харків: УкрДАЗТ, 2007.

4. Дядюн С.В., Костенко О.Б., Меркулов В.С., Філіппенко І.Г., Шумеєв В.В. Збірник задач та прикладів з обчислювальної техніки та програмування. – Харків: ХДАМГ, 2001. – 55 с.

5. Завдання до контрольної роботи з дисциплини «Обчислювальна техніка та програмування». – В 2-х ч. – Харків: ХарДАЗТ, 2002.

6. Информатика: Задачник-практикум /Под ред. И.Г.Семакина, Е.К.Хеннера. – В 2-х т. – М.: Лаборатория базовых знаний, 2000.

7. Інформатика, комп'ютерна техніка, комп'ютерні технології: Посібник / За ред. О.І. Пушкаря. – К: Академія, 2001

8. Кетков Ю.Л. GW-, Turbo- и Quick-Basic для IBM PC. – М.: Финансы и статистика, 1992. – 240 с.

9. Мамонтов Д.В. Quick Basic в задачах и примерах. – С.Пб.: Питер, 2006. – 256 с.

10. Мельникова О.И., Банюшкина А.Ю. Начала программирования на языке QBasic: Учеб. пособие. – М.: ЭКОМ, 1997. – 304 с.

11. Основи алгоритмізації базових обчислювальних процесів: Навч. посібник / В.С.Меркулов, В.О.Гончаров, І.Г.Бізюк, В.М.Бутенко, О.В. Головко. – Харків: УкрДАЗТ, 2008. – 163 с.

12. Петров А.В., Алексеев В.Е. Вычислительная техника и программирование. – М.: Высшая школа, 1990. – 480 с.

13. Райтингер М., Муч Г. Visual Basic 6.0. – К: BHV, 2000. – 288 с.

14. Семакин И.Г., Вараксин Г.С. Информатика: Структурированный конспект базового курса. – М.: Лаборатория базовых знаний, 2001.

15. Уолш Б. Программирование на БЕЙСИКе. – М.: Радио и связь, 1988. – 120 с.

16. Филиппенко И.Г. Программирование на языке БЕЙСИК: Учеб. пособие. – Харьков: ХИИТ, 1988.

17. Филиппенко И.Г., Карасюк В.В., Карачаров А.Ф., Гончаров В.А., Меркулов В.С., Гвозденко М.В. Методические указания к лабораторным работам №7-12 по дисциплине «Вычислительная техника и програмирование» Ч. 2. Язык Бейсик для ПЭВМ ДВК-2М. – Харьков: ХИИТ, 1992.

18. Філіппенко І.Г., Гончаров В.О., Меркулов В.С. Основи побудови ПК: Конспект лекцій з дисципліни «Обчислювальна техніка та програмування». – Харків: УкрДАЗТ, 2005. – Ч. 1.

19. Філіппенко І.Г., Гончаров В.О., Меркулов В.С. Основи алгоритмізації: Конспект лекцій з дисципліни «Обчислювальна техніка та програмування». – Харків: УкрДАЗТ, 2005. – Ч. 2.

20. Філіппенко І.Г., Гончаров В.О., Меркулов В.С. Програмування інженерно-технічних задач в середовищі QBasic: Конспект лекцій з дисципліни «Обчислювальна техніка і програмування». – Харків: УкрДАЗТ, 2007. – Ч. 3. – 134 с.

21. Філіппенко І.Г., Карачаров А.Ф., Бантюков С.Є., Гончаров В.О., Меркулов В.С., Конопацька В.М., Міщенко К.О. Методичні вказівки до лабораторних робіт з дисциплін «Основи інформатики», «Обчислювальна техніка та програмування для ЕОМ». Ч. 1. Основи проектування алгоритмів. – Харків: ХарДАЗТ, 2002.

22. Філіппенко І.Г., Бантюков С.Є., Конопацька В.М., Шумеєв В.В. Методичні вказівки до лабораторних робіт з дисциплін «Основи інформатики», «Обчислювальна техніка та програмування». Ч. 2, 3. Програмування мовою BASIC. – Харків: ХарДАЗТ, 2000.

23. Филичев С.В. Занимательный Basic: Практическое пособие. – М.: ЭКОМ, 1997. – 192 с.

24. Visual Basic 6.0. – С.Пб.: BHV Питер, 1999. – 992 с.