ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра обчислювальної техніки і систем управління

В.Г. Пчолін, О.Є. Пєнкіна

ОСНОВИ СУЧАСНОЇ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

Конспект лекцій

з дисципліни

«ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ»

Харків – 2015

Пчолін В.Г., Пєнкіна О.Є. Основи сучасної технології програмування: Конспект лекцій. − Харків: УкрДАЗТ, 2015. – 30 с.

Даний конспект лекцій розроблено відповідно до програми курсу „Інформаційні системи та технології”. Матеріал, викладений у конспекті лекцій, охоплює розділи, необхідні студентам для опанування роботи на персональному комп’ютері у середовищі QBASIC, зокрема основи програмування та прийоми відлагодження основних алгоритмічних структур.

Конспект лекцій містить приклади програм або їх фрагменти, які в подальшому можуть використовуватись при розрахунках і моделюванні економічних задач.

Рекомендується для студентів економічного факультету всіх форм навчання напряму підготовки 6.030601 „Менеджмент”.

Іл. 3, табл. 11, бібліогр.: 4 назв.

Конспект лекцій розглянуто і рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 31 березня 2014 р., протокол № 9.

Рецензент

проф. В.Л. Дикань

Пчолін В.Г., Пєнкіна О.Є.

ОСНОВИ СУЧАСНОЇ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

Конспект лекцій

з дисципліни

«Інформаційні системи та технології»

Відповідальний за випуск Пєнкіна О.Є.

Редактор Буранова Н.В.

Підписано до друку 22.04.14 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 1,00. Тираж 50. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту, 61050, Харків-50, майдан Фейєрбаха, 7.

Свідоцтво суб’єкта видавничої справи ДК № 2874 від 12.06.2007 р.

УКРАЇНСЬКА ДЕРЖАВНА АКАДЕМІЯ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра „Обчислювальна техніка та системи управління ”

ОСНОВИ СУЧАСНОЇ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«Інформаційні системи та технології»

для студентів факультету економіки транспорту

Харків 2014

Пчолін В.Г., Пєнкіна О.Є. Основи сучасної технології програмування: Конспект лекцій. − Харків: УкрДАЗТ, 2015. – 30 с.

Даний конспект лекцій розроблено відповідно до програми курсу „Інформаційні системи та технології”. Матеріал, викладений у конспекті лекцій, охоплює розділи, необхідні студентам для опанування роботи на персональному комп’ютері у середовищі QBASIC, зокрема основи програмування та прийоми відлагодження основних алгоритмічних структур.

Конспект лекцій містить приклади програм або їх фрагменти, які в подальшому можуть використовуватись при розрахунках і моделюванні економічних задач.

Рекомендується для студентів економічного факультету всіх форм навчання напряму підготовки 6.030601 „Менеджмент”.

Іл. 3, табл. 11, бібліогр: 4 назв.

Конспект лекцій розглянуто і рекомендовано до друку на засіданні кафедри “Обчислювальна техніка та системи управління” 31 березня 2014 р., протокол № 9.

Рецензент
проф. В.Л. Дикань

ЗМІСТ

ВСТУП

Написання хороших програм
вимагає розуму, смаку і терпіння.
Б. Страуструп

Basic (Бейсік) − універсальний код символічних інструкцій для початківців. Був розроблений у 1963 році професорами Дартмутського коледжу Томасом Курцем і Джоном Кемені. 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).

Простота граматики й синтаксису, легкість освоєння зробили Basic популярним серед користувачів-непрофесіоналів для математичних і науково-технічних розрахунків. Тому він і досі широко застосовується для навчання основам програмування і для розв’язання відносно простих задач програмування.

Одним з найбільш поширених варіантів мови Бейсік стала реалізація інтерпретатора QBASIC. Особливо слід було б виділити досить потужні графічні та звукові можливості QBASIC, що дають змогу програмувати цією мовою ігри. QBASIC – це інтегрована система програмування, що має власну керуючу оболонку, редактор текстів програм мовою програмування QBASIC, засоби пуску, налагодження й прихованого складання програм, потужний електронний довідник з системи.

СИСТЕМА ПРОГРАМУВАННЯ QBASIC

1 Синтаксис мови

У тексті будь-якою природною мовою можна виділити чотири основні елементи: символи, слова, словосполучення і пропозиції. Подібні елементи містить і алгоритмічна мова, тільки слова називають лексемами (елементарними конструкціями), словосполучення − виразами, а пропозиції − операторами. Лексеми утворюються із символів, вирази − з лексем і символів, а оператори − із символів, виразів і лексем:

- ✓ алфавіт мови, або його символи, − це основні неподільні знаки, за допомогою яких пишуться всі тексти мовою;
- ✓ лексема, чи елементарна конструкція, − мінімальна одиниця мови, що має самостійний зміст;
- ✓ вираз задає правило обчислення деякого значення;
- ✓ оператор задає закінчений опис деякої дії;
- ✓ функція задає виконання однакових обчислювальних операцій.

1.1 Алфавіт

Основою будь-якої мови програмування є ***алфавіт*** − набір допустимих знаків, які можна використовувати для запису програм.

Алфавіт мови QBASIC включає всі відображувані і 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).

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

Наприклад, символ BEL з кодом 7 − «висновок звукового сигналу», символ HT з кодом 9 − «горизонтальна табуляція», символ LF з кодом 10 − «переклад рядка», символ CR з кодом 13 − «повернення каретки».

Усі символи можна поділити на групи:

- ✓ латинські літери (A- Z, a - z);
- ✓ російські літери (А - Я, а- я);
- ✓ цифри (0-9);
- ✓ розділові символи (обмежувачі) (див таблицю 1.1-1);

Таблиця 1.1-1

.	десяткова крапка	**,**	кома
:	двокрапка	**"**	лапки
( )	круглі дужки	**_**	пропуск
;	крапка з комою	**'**	одинарні лапки (апостроф)

- ✓ спеціальні знаки (див. таблицю 1.1-2).

Таблиця 1.1-2

$	знак грошової одиниці	**_**	символ підкреслення
&	комерційне І (амперсанд)	**%**	знак відсотка
#	номер	**? ! {}...**	та інші
@	комерційне АТ		

1.2 Команди, оператори

Команди і оператори − це синтаксичні конструкції мови програмування, їм відповідають певні підпрограми обробки інформації. Команди оперують усією програмою або її частиною. Оператори служать для реалізації алгоритму програми. Команди і оператори − зарезервовані слова мови програмування, вони не можуть бути використані в якості змінних або міток. У командах і операторах програм використовуються тільки символи латинського алфавіту. Російські літери використовуються тільки для запису символьних констант і коментарів.

QBASIC не робить різниці між великими і малими латинськими літерами в службових словах, позначеннях змінних і інших об'єктів.

За призначенням оператори поділяють на такі групи:

- ✓ описові оператори − для опису даних, типів змінних, розмірів масивів, нестандартних функцій тощо;
- ✓ оператори присвоювання − для задання початкового значення або зміни поточного значення змінної;
- ✓ оператори введення-виведення даних − для введення і виведення інформації;
- ✓ оператори керування процесом обробки інформації: оператори переходу, організації розгалужень і циклів тощо;

- ✓ інші оператори, що забезпечують додаткові можливості: оператори для роботи з файлами даних, оператори для графічних побудов, одержання звукових ефектів тощо.

Оператори складаються з нероздільних елементів мови: службових слів, чисел, символів операцій. Мова QBASIC налічує більше 200 типів операторів (див таблицю 1.2-1).

Таблиця 1.2-3

DATA – дані	PLAY − грати
DEF FN – визначення функції	OPTION BASE – задати базу
DIM – розмір	PRINT – друкувати
FOR-TO-STEP – для – до – крок	REM – пояснення
GOTO – перейти до	RESTORE – оновити
IF-THEN – якщо – то	RETURN – повернутися
INPUT – ввести	STOP – зупинитися
NEXT – наступний	END − кінець

1.3 Константи та змінні

При обробці інформації програми оперують з даними. Залежно від способу подання даних вони поділяються на ***константи*** та ***змінні***.

Константи − дані , значення яких не змінюються при виконанні програми.

Константи за типом поділяються:

- ✓ на числові;
- ✓ логічні;
- ✓ символьні.

Числові константи за відсутності або наявності дробової частини і за способом зберігання в пам'яті ЕОМ поділяються на цілі і дійсні (див. подання даних у таблиці 1.3-1).

Змінна − це іменована область пам'яті, в якій зберігаються дані певного типу. У змінної є ім'я і значення. Ім'я служить для звернення до області пам'яті, в якій зберігається значення. Під час виконання програми значення змінної можна змінювати і присвоювати значення констант, інших змінних, результатів обчислень.

Таблиця 1.3-4

Константа	Формат	Приклад
Цілі	Послідовність десяткових цифр і спеціальних символів зі знаком або без нього	123%, -89%. 19926%,-
Дійсні	Десятковий: послідовність десяткових цифр зі знаком і точкою, яка розділяє цілу і дробові частини Експонентний: у вигляді мантиси і порядку. Мантиса записується аналогічно запису відповідної дійсності константи, після цього	45.23, -.6, ,29.5! 0.2E6, 1.3E-04.11e-3, 5E100, 0.56E+12, 6.345D-2
Логічні	Істина: Хибність:	TRUE FALSE
Символьні	Узята в лапки послідовність символів (букв, цифр, спеціальних знаків − усіх символів, які можна набрати на клавіатурі). Максимальна довжина символьної константи обмежена довжиною програмного рядка (255 символів)	"Харків", " аудиторія 2.221", „=Ю?*+", ”777”

Кожна змінна позначається унікальним ім'ям − ***ідентифікатором***.

Ідентифікатор може містити:

- ✓ алфавітно-цифрові символи (латинські літери);
- ✓ знак підкреслення;
- ✓ першим символом повинна бути не цифра;
- ✓ ім'я змінної не повинно збігатися з ключовими словами QBASIC.

Необхідно чітко уявляти, що кожному ідентифікатору в пам'яті ЕОМ відповідає цілком певна адреса комірки, і зміна значення змінної зводиться до зміни вмісту відповідної області пам'яті.

Наприклад: A, b12%, C_mod&, time#, Name$.

Тип змінної може бути зазначений:

- ✓ ***явно*** − за допомогою спеціального символа (%, &,!, #, $), який завершує ім'я змінної (див. таблицю 1.3-2);

Таблиця 1.3-5

Тип	Довжина, байт (біт)	Діапазон значень	Спеціальній символ
Цілі (INTEGER)	2(16)	від -32768 до 32767	%
Цілі (LONG)	4(32)	від -21474836478 до +21474836478	&
Дійсні (SINGLE)	4(32)	від 3.4e-38 до 3.4e+38	без спеціальних символів або **!**
Дійсні(DOUBLE)	8(64)	від 1.7e-308 до 1.7e+308	#
Символьні(STRING)	1(8)	до 256	$

✓ ***неявно*** − за належністю першого символа ідентифікатора заданому діапазону літер.

Неявне оголошення типу змінної здійснюється за допомогою операторів (див. таблицю 1.3-3).

Таблиця 1.3-6

DEFINT	Двобайтовий цілочисельний
DEFLNG	Чотирибайтовий цілочисельний
DEFSNG	Чотирибайтовий дійсний
DEFDBL	Восьмибайтовий дійсний
DEFSTR	Символьний тип

Усі 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. Використане у форматі позначення <> означає обов'язковий елемент конструкції.

1.4 Функції

При розв’язанні математичних завдань і обробці символьних даних часто доводиться виконувати однакові обчислювальні операції. Тому в системах програмування є велика кількість вбудованих підпрограм, що виконують ці операції. Ці

підпрограми називають функціями. Крім того обчислювальні системи надають користувачеві можливість створювати свої функції користувача для реалізації обчислювальних операцій, що часто використовуються. Синтаксис функції:

Ім'я_функції (аргумент [, аргумент])

Увага! *У функціях аргументи записуються у круглих дужках.*

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

- ✓ математичні функції;
- ✓ числові функції символьних аргументів;
- ✓ символьні функції;
- ✓ функції введення, виведення і доступу до пам'яті;
- ✓ системні змінні.

Основні *математичні функції* мови програмування QBASIC наведені у таблиці 1.4.-1.

Таблиця 1.4-1

Функція		**Опис**
Математика	**QBasic**	
Арифметичні функції		
| x |	ABS(x)	Обчислює абсолютне значення числа x
[x]	CEIL(x)	Повертає ціле число, яке є більшим або дорівнює x для додатних чисел і є меншим або дорівнює x для від'ємних чисел
] x [	INT(x)	Повертає ціле число, яке є меншим або дорівнює x як для додатних, так і для від'ємних чисел
e^x	EXP(x)	Повертає число e , піднесене до вказаного степеня
ln x	LOG(x)	Обчислює натуральний логарифм аргументу
r n d(N)	RND(N)	Повертає випадкове число в інтервалі від 0 до 1. При N<0 повертає певне число, залежне від N; при N=0 − псевдовипадкове число; при N>0 − нове випадкове число

Функція		Опис
Математика	**QBasic**	
Округлювання чисел	ROUND(x,n)	Повертає число, округлене до заданого числа десяткових знаків

Продовження таблиці 1.4–1

Функція		Опис
Математика	**QBasic**	
Арифметичні функції		
Знак числа	SGN(x)	Повертає знак числа: 1, якщо x>0; 0, якщо x=0; -1, якщо x<0
$\sqrt{x}$	SQR(x)	Обчислює корінь квадратний з аргументу
Тригонометричні та зворотні тригонометричні функції		
cos x	COS(x)	Обчислення косинуса кута
sin x	SIN(x)	Обчислення синуса кута
tg x	TAN(x)	Обчислення тангенса кута
arctg x	ATN(x)	Обчислення арктангенса кута

$\mathrm{Log_a(x)} = \frac{\mathrm{Log_b(x)}}{\mathrm{Log_b(a)}}$	$\arccos(x) = \pi/2 - \operatorname{arctg}(x/\sqrt{1-x^2})$
$\arcsin(x) = \operatorname{arctg}(x/\sqrt{1-x^2})$	$\operatorname{arcctg}(x) = \pi/2 - \operatorname{arctg}(x)$

Числові функції символьних аргументів

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

ASC, CVI, CVS, CVD, INSTR, LEN, VAL і ін.

Символьні функції

Значеннями цих функцій є ланцюжки символів. ***CHR$, LEFT$, RIGHT$, MID$, STRING$, LTRIM$*** і ін.

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

Функції введення-виведення і доступу до пам'яті

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

CSRLIN − повертає номер рядка поточного положення курсору.

FRE − обсяг вільної частини пам'яті робочої області QBASIC.

POINT − координати точки екрана.

PEEK − вміст байта пам'яті і ін.

Системні змінні

TIME$ − системний час.

DATE$ − системна дата і ін.

Повністю список усіх функцій можна знайти у довідці QBASIC і системній документації.

1.5 Вирази

Вирази являють собою комбінацію змінних і операндів, з'єднаних знаками відносин. Залежно від знаків операцій, що використовуються, розрізняють арифметичні вирази, логічні вислови й рядкові вирази. Синтаксис усіх виразів однаковий:

< операнд > знак_операції < операнд >

Як операнди у виразах можуть використовуватися константи, змінні, функції та інші вирази.

Арифметичні вирази

Знаки арифметичних операцій наведені в таблиці **1.5-1**

Таблиця 1.5-1

Знак операції	**Приклад**	**Опис**
-	-a	Зміна знака числа
^	y=x^a	Піднесення числа до степеня
*, /	y=a*b/c	Множення і ділення чисел
Mod	y = a Mod b	Повертає залишок при цілому розподілі двох чисел
+, -	y=a+b-c	Додавання і віднімання чисел
Порядок виконання операцій може бути змінений шляхом використання дужок. Вирази в дужках виконуються в першу чергу		

( )	y=(a+ b)/c	Дужки

Логічні вирази

Логічні вирази складаються з логічних операцій (див. таблицю 1.5.-2) і логічних відношень. У них наявні два операнди, які QBASIC розглядає як 16-річні бінарні ланцюжки, над якими зліва направо виконуються дії за допомогою логічних операторів (див. таблицю 1.5.-3)

Таблиця 1.5-2

Знак операції	***Приклад***	***Опис***
=	a = b	дорівнює
<	a < b	менше
>	b > c	більше
<=	a <= d	менше або дорівнює
>=	c >= e	більше або дорівнює
<>	f <> g	не дорівнює

Таблиця 1.5-3

Оператор	***Призначення***	***Приклад***			
		Вираз	***A***	***B***	***Результат***
And	Логічне множення (I)	A And B	1 1 0 0	1 0 1 0	1 0 0 0
Eqv	Перевірка логічної еквівалентності (=)	A Eqv B	1 1 0 0	1 0 1 0	1 0 0 1
Imp	Імплікація	A Imp B	1 1 0 0	1 0 1 0	1 0 1 1
Not	Операція логічного заперечення (Немає)	NOT A	1 0		0 1

Оператор	***Призначення***	***Приклад***			
		Вираз	***A***	***B***	***Результат***
Or	Операція логічного додавання (АБО)	A Or B	1 1 0 0	1 0 1 0	1 1 1 0
Xor	Операція виключає АБО	A Xor B	1 1 0 0	1 0 1 0	1 0 0 1

Усі оператори повертають значення "істина" (не-нуль) або "хибність" (нуль), що використовуються при ухваленні рішення про подальший хід обчислювального процесу.

Рядкові вирази

У символьних виразах можуть використовуватися тільки операції додавання (конкатенації). Як знаки операцій можуть використовуватися символи + , - і знак & (амперсенд). Знаки "+" і "&" рівносильні. Як операнди в символьних виразах можуть бути тільки рядки символів. Результатом операції також є рядок символів.

2 Середовище програмування QBASIC

2.1 Завантаження програми

Завантаження QBASIC здійснюється з відповідного каталогу вибором файлу *qbasic.exe*. Після запуску програми на екрані дисплея з'являється робочий екран (рисунок 2.1-1).

Тепер можна натиснути клавішу **[*Enter*]** або **[*Esc*]**. При натисненні клавіші [***Enter***] викликається провідник для користувача (рисунок 2.1-2).

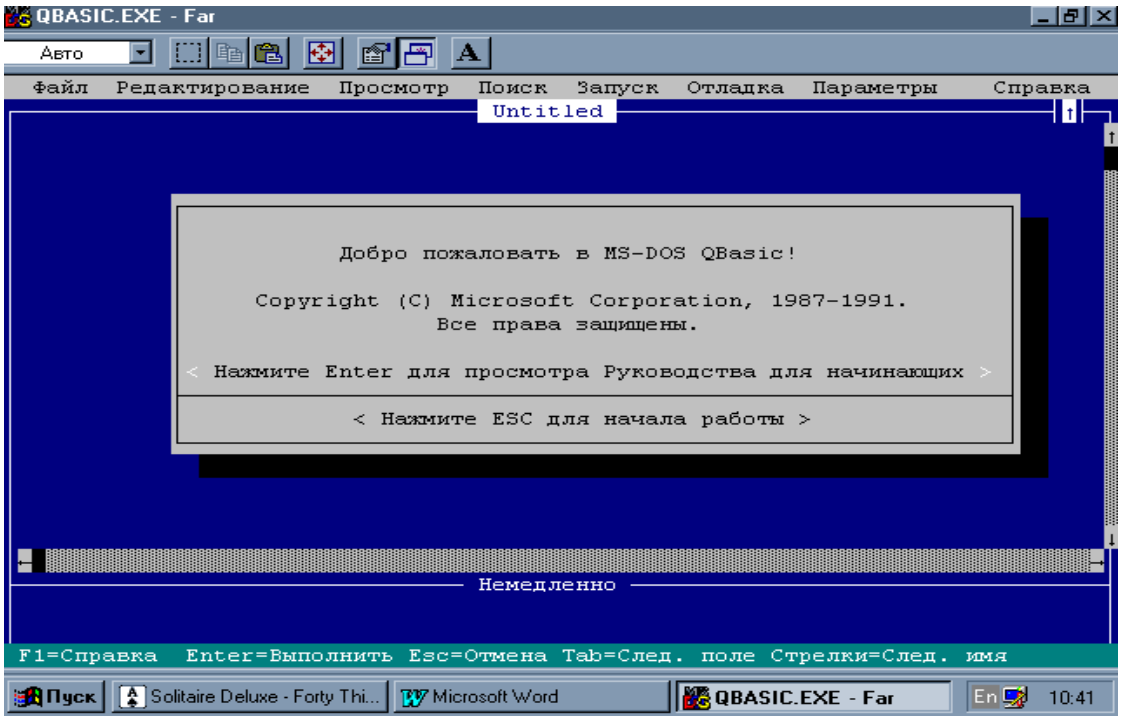

Рисунок 2.1-1 - Перше вікно при запуску QBASIC

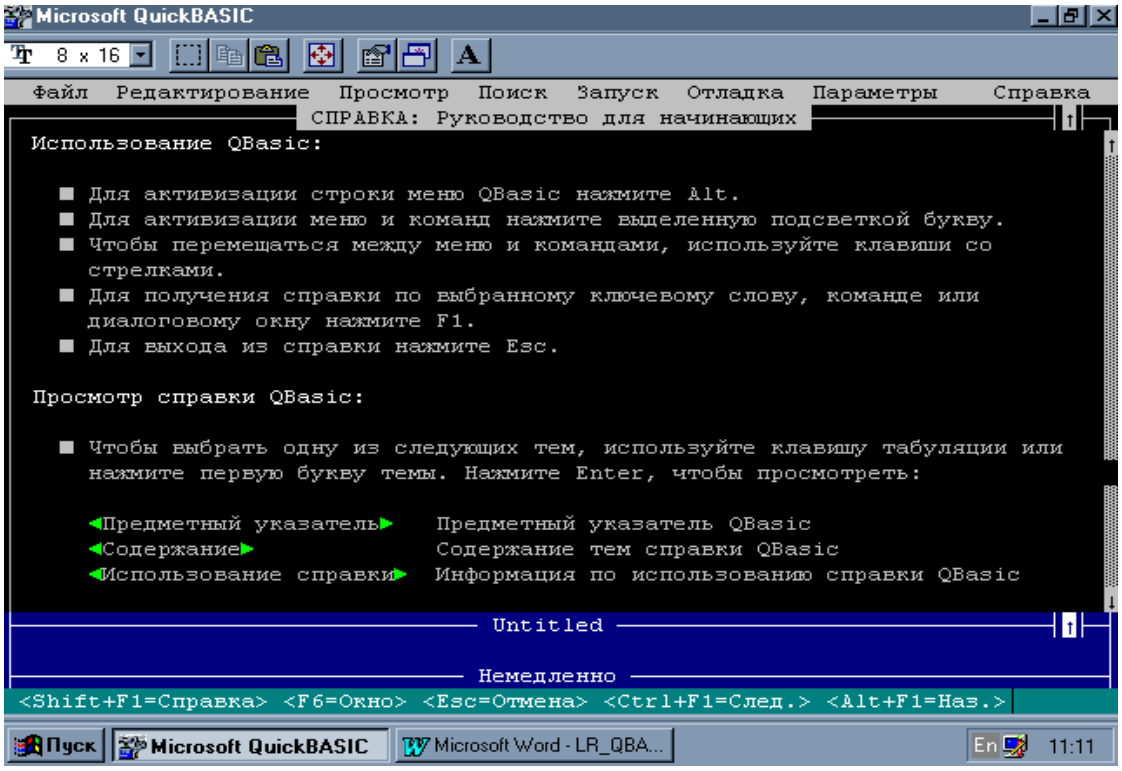

Рисунок 2.1-2 - Провідник

2.2 Система меню

У верхній частині екрана розташовано головне меню QBASIC, яке містить назви основних функцій середовища QBASIC.

По черзі розкрити пункти головного меню і за допомогою рядка підказки та системи довідок ознайомитись і законспектувати виклик основних функцій середовища QBASIC [***File*** (Файл), ***Edit*** (Редагування), ***Run*** (Запуск), ***Help*** (Довідка)].

2.2.1 Введення та редагування програм

Головне вікно QBASIC розподілено на дві основні частини − вікно редагування та вікно безпосереднього виконання.

Вікно редагування

На початку роботи QBASIC розташовує курсор у вікні редагування. Це дозволяє вводити текст нової програми (команда ***New*** з меню ***File***), завантажувати у це вікно тексти існуючих програм (команда **Open** з меню ***File***), редагувати введені тексти, запускати програми на виконання (команда ***Start*** з меню ***Run***), зберігати їх у файлах на диску (команда ***Save*** з меню ***File***).

Вікно безпосереднього виконання

Це вікно розташовується у нижній частині екрана. У цьому вікні можна безпосередньо отримувати результати виконання команди після її введення та натиснення клавіші **[*Enter*]**. Перехід у це вікно здійснюється після натиснення клавіші **[*F6*]**.

Редактор QBASIC

Розглянемо роботу редактора на прикладі простої програми, що містить один оператор.

PRINT ’’УкрДАЗТ’’

Закінчивши набір оператора, натисніть клавішу **[*Enter*]**. Якщо ви допустили помилку при введенні оператора, її можна виправити, використовуючи клавіші переміщення курсору та клавіші вилучення символа:

[*Delete*] − вилучає символ над курсором;

[*Backspace*] − вилучає символ зліва від курсору.

2.2.2 Запуск програми на виконання та перегляд результатів

Для виконання введеної програми треба викликати команду ***Start*** з меню ***Run*** або використати комбінацію клавіш ***Shift+F5***.

На екран буде виведений результат виконання програми та повідомлення «*Press any key to continue*» (для продовження натисніть будь-яку клавішу).

2.2.3 Збереження програми на диску

Після завершення роботи з програмою її треба зберегти на диску у вигляді файлу. Це дає змогу повернутись до роботи із збереженою програмою у майбутньому.

Для збереження програми треба виконати такі дії:

- ✓ виберіть опцію меню ***File*;**
- ✓ виберіть команду ***Save*** (Зберегти) і натисніть клавішу [***Enter***];
- ✓ якщо програма ще не має імені, то вона у середовищі QBASIC буде помічена як ***Untitled***. Введіть ім'я файлу, у якому ви бажаєте зберегти свою програму, та натисніть клавішу **[*Enter*].** Вікно збереження файлу показано на рисунку 2.2.3-1.

Увага! *Ім'я файлу має бути унікальним.*

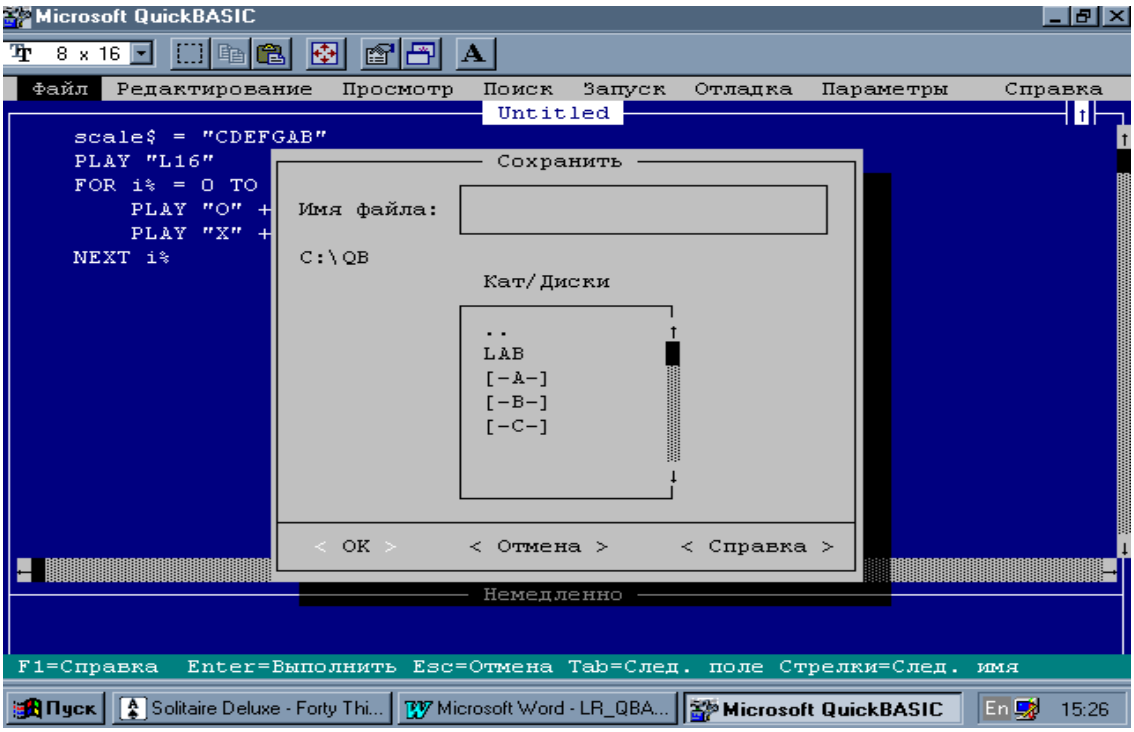

Рисунок 2.2.3-1 - Вікно збереження файлу

2.2.4 Вихід із середовища QBASIC

Для виходу з середовища QBASIC виконайте послідовно такі дії:

- ✓ перейдіть у головне меню та виберіть опцію ***File*** (Файл);
- ✓ виберіть команду ***Exit*** (Вихід);
- ✓ натисніть клавішу [***Enter***].

3 Програмування в середовищі QBASIC

3.1 Структура QBASIC програм

Програма набирається у вікні програми порядково. Довжина програмного рядка не повинна перевищувати 255 символів. Автоматичного переходу на інший рядок не передбачено. Тому якщо довжина рядка перевищує 80 символів, то текст програми зсувається вліво і зникає з екрана. Допускається переносити довгі рядки. Для цього в кінці першої частини рядка вставляється пробіл і символ підкреслення "_". Дозволяється розміщувати в одному рядку кілька операторів, розділяючи їх двокрапкою. Рядок завершується символом повернення каретки, який автоматично вводиться при натисканні клавіші *Enter*.

QBasic не вимагає обов'язкової нумерації рядків. Для зміни процесу обчислень у програмі використовуються мітки. Ім'я мітки формується за тими ж правилами, що й ім'я змінної. Ім'я мітки закінчується двокрапкою, наприклад: M1 : , V1 : , Pause1: і т. д. В одному рядку з міткою не допускається записувати виконувані оператори. Допускається включати не виконувані оператори та коментарі.

Щоб зробити програму більш зрозумілою, в неї рекомендується включати коментарі. Правіше коментаря не можна включати жодних операторів. Коментарі вводяться оператором *REM* або апострофом − символ ('). У рядку коментаря дозволяється використовувати будь-які символи.

Приклад програми.

REM процедура обчислення суми кінцевого ряду
' Табулювання функції однієї змінної
' Пошук мінімуму функції однієї змінної
INPUT "Вкажіть кордону A і B інтервал табулювання функції", A, B

```
dX = 0.1 ' крок табуляції
Max = - 1E38 ' початкове привласнення
FOR x = A TO B STEP dX ' заголовок циклу
y = EXP ( x ) * SIN ( x ^ 2 )
IF y > Max THEN Max = y
NEXT x
PRINT "Max =" ; Max
END
```

3.2 Лінійні програми

Лінійні програми являють собою послідовність операторів, циклів і умовних переходів немає. Структурна схема такої програми включає блок введення даних, блок обчислень і блок виведення результатів.

3.2.1 Оператори введення даних

Для введення даних використовуються оператори ***LET, DATA, READ, RESTORE, INPUT, LINE INPUT,*** *функція* ***INPUT$.***

Оператор ***LET*** служить для присвоєння значень змінним.

Формат оператора:

LET <ім'я змінної> = < вираз>

Оператор *LET* може бути опущений, тому вирази

LET A = exp (x) + sin (x) і A = exp (x) + sin (x) **еквівалентні.**

Оператори ***DATA*** і ***READ*** дозволяють вводити дані з програми. Оператор ***DATA*** заносить дані в спеціальну область оперативної пам'яті комп'ютера, а оператор ***READ*** зчитує ці дані з оперативної пам'яті і привласнює їх змінним. Ця область даних містить спеціальний покажчик. При запуску програми покажчик встановлюється на початку області даних. При зчитуванні даних з області пам'яті він також переміщається. Якщо число змінних в операторі ***READ*** більше числа значень в операторі ***DATA***, то покажчик виходить за область даних і буде видано повідомлення про помилку. Типи даних в операторах ***DATA*** і ***READ*** мають

відповідати один одному. Ці оператори можуть розташовуватися в будь-якому місці програми. Рядкові змінні записуються без лапок , якщо в них немає пробілів, і беруться в лапки , якщо містять пробіли.

Формати операторів:

DATA <список даних>
READ <список змінних>

Наприклад:
DATA 125, 34.78, 1.24E-5, „Харків”, “Перша столиця”
READ A,B,D,C$,C1$

Змінним A, B і D будуть присвоєні, відповідно, числові значення 125, 34.78 і 0.0000124, змінним C$ і C1$ − *„Харків”* і *“Перша столиц*я"

Для повторного використання даних застосовується оператор ***RESTORE***. Формат оператора:
RESTORE [<***мітка***>].

Якщо мітка в операторі ***RESTORE*** не вказана, то покажчик встановлюється в початок області даних, інакше покажчик встановлюється на відповідну мітку в області даних.

Оператор INPUT служить для введення даних з клавіатури в режимі діалогу з користувачем. Формат оператора:

INPUT [;] " текстовий вираз " [;/ ,] <список змінних >

де [;] − необов'язковий параметр, залишає курсор у поточному рядку;

[;/ ,] − означає, що як роздільник може використовуватися один із вказаних знаків " ; " або " , " .

При виконанні оператора ***INPUT*** на екран видається запит на введення даних.

Якщо як роздільник використовується " ; " , то запит супроводжується виведенням на екран знака питання, при використанні як роздільника " , " знак питання на екран не виводиться. Текстове повідомлення дозволяє зробити запит зрозумілим користувачеві. У списку змінних дані розділяються комами.

Наприклад:

INPUT " Введіть змінні A і B"; A , B
INPUT " Введіть Ваш рік народження ", GR $

де A і B − числові змінні, GR$ − символьна змінна.

Оператор ***LINE INPUT*** служить для введення однієї символьної змінної. При введенні значення символьна змінна в лапки не береться, в ній допускається наявність пробілів та інших роздільників. Кінець рядка визначається символом повернення каретки (натискання клавіші ***ENTER***). Формат оператора ***LINE INPUT*** аналогічний формату оператора ***INPUT*** Функція ***INPUT$*** служить для введення символів, які не відображаються на екрані, наприклад пароля. Формат функції:

c1$ = INPUT$ (n [, [#] nf])

де n − число символів, що вводять; # − номер каналу при введенні даних з файлу; nf − ім'я файлу.

Наприклад:

LINE INPUT " Введіть прізвище , ім'я та по батькові " , FIO $
B$ = INPUT$ (5)

Символьній змінній ***B$*** присвоюється код із п'яти символів, при видачі запиту введені символи відображатися на екрані не будуть. Коли в програмі зустрічається оператор ***INPUT$ (n***), програма зупиняється і чекає введення даних. Тому даний оператор у форматі ***B$ = INPUT$ (1)*** може бути використаний для зупинки програми до натискання будь-якої клавіші.

3.2.2 Оператори виведення даних

Для виведення даних використовуються оператори ***PRINT*** і ***LPRINT***, ***PRINT USING*** і ***LPRINT USING***.

Оператор ***PRINT*** виводить дані на екран монітора, а ***LPRINT*** − на принтер. Обидва оператори мають однаковий формат. Оператори ***PRINT USING*** і ***LPRINT USING*** мають те саме призначення, але додатково дають змогу форматувати дані, що виводяться.

Формат оператора ***PRINT***:

PRINT "Коментар" [;/ ,] <список виразів > [;/ ,]

Коментар служить для пояснення виведеної інформації; символи **[;/ _ / ,]** − роздільники.

Якщо як роздільник використовується символ [;] або пробіл, то текст, що виводиться, розміщується один біля одного без пробілів, при виведенні чисел одна позиція резервується для знака числа.

Якщо як роздільник використовується [,], то кожне нове повідомлення друкується в новій зоні. Оператор ***PRINT*** формує рядок виводу, який розбитий на 5 зон по 14 символів. Це дозволяє розміщувати інформацію, що виводиться, по колонках.

У список виразів може поміщатися будь-яка інформація. Виведені вирази списку розділяються комами.

В кінці рядка можуть бути розташовані керуючі символи: крапка з комою чи кома . За наявності цих керуючих символів курсор залишається в поточній позиції, і наступний оператор ***PRINT*** буде виводити інформацію з цієї позиції. Діють ці керуючі символи так само, як і керуючі символи після коментаря.

Наприклад:

```
PRINT " X=" ; x                    ' виводиться значення x
PRINT " X=" ; x,"Y =" ; y          'виводяться значення x і y
PRINT " X=" x,"Y =";SIN(x)         'виводиться значення x,
                                   обчислюється і
                                   'виводиться значення функції SIN (x)
```

Для керування виведенням інформації в список виразів оператора ***PRINT*** можуть включатися функції ***TAB*** і ***SPC*** .

Функції ***TAB*** і ***SPC*** переміщують точку виведення на задане число позицій. Відмінність у використанні функцій ***TAB*** і ***SPC*** полягає в тому, що функція ***TAB*** переміщує точку введення від краю екрана, а функція ***SPC*** − відносно поточної позиції.

Оператор ***PRINT USING*** призначений для виведення інформації на екран з форматуванням.

Формат оператора ***PRINT USING***:

PRINT USING { C$ | e$ } ; <список змінних >

де ***C$ і e$*** − шаблони для виведення символьної або числової інформації.

Шаблони для виведення числової інформації.

Якщо число знаків, зазначених у цілій частині числа шаблону, менше, ніж фактичне, то число виводиться на екран повністю, але перед першим символом числа ставиться знак %.

Оператори ***LPRINT*** і ***LPRINT USING*** мають той самий синтаксис, що й оператори ***PRINT*** і ***PRINT USING*** (див. таблицю 3.2.2.-1), але виводять інформацію на друкувальний пристрій.

Таблиця 3.2.2-1

Символ	**Опис**	**Приклад**
#	Резервує знакомісце для числа, пробілу, знака $ або *	PRINT USING "#####"; a,b,c – зарезервовано п’ять знакомісць при виведенні числа на екран монітора
.	Позиція десяткового дробу	PRINT USING "##.###"; a,b,c – числа a , b , c будуть виведені на екран з трьома знаками після десяткової точки
^^^^	Місце розташування порядку числа (E + pp) (E -pp)	PRINT USING "#.####E+pp"; a,b,c – числа будуть виведені в експоненційній формі
±	- виводить знак числа. Може розташовуватися на початку або в кінці шаблону	PRINT USING "-##.###"; a,b ,c – перед від’ємними числами буде виводитися знак числа
**	Заповнення лідируючих пробілів символами "*"	PRINT USING "**###.##"; a,b,c – якщо число знаків у числі менше, ніж число знакомісць, зазначених у шаблоні, то лідируючі пробіли заповнюються зірочками
$	Виведення знака $ перед старшої значущою цифрою	PRINT USING "$###.##"; a,b,c − перед старшою значущою цифрою буде виведений знак $
,	Роздільник розрядів числа	PRINT USING "###,###,###.##"; a,b,c – ціла частина числа розділяється на тріади від молодшого розряду до старшого, а дрібна частина на тріади не

		поділяється

Наприклад:
PRINT USING "# # # # # . # # " ; 123.6787 123.68
PRINT USING "# # # # # . # # " ; -123.6787 -123.68
PRINT USING " + # . # # " ; 123.6787 %+123.68
PRINT USING "#. # # ^ ^ ^ ^ " ; .01236787#0.12D – 01
*PRINT USING " ** # # # # # . # # " ; 123.6787 **** 123.68*
PRINT USING " $ # # # # . # # " ; 123.6787 $123.68

Як приклад програми лінійного обчислювального процесу розглянемо задачу.

Розрахувати, яку суму грошей на рахунку потрібно мати клієнту фінансової установи для того, щоб через 3 роки отримати 20000 грн, якщо відсоткова ставка дорівнює 14 % річних?

Формула для розрахунку дисконтованого грошового потоку:

$$PV = \frac{FV}{(1+\frac{pr}{100})^t},$$

де PV − поточна сума;
FV − майбутня сума;
pr − ставка дисконтування;
t − кількість років.

```
CLS                    ' очищення екрана виводу
REM Обчислення суми вкладу
t=3                    ‘ термін вкладу
pr=14                  ’ річна ставка
FV=20000               ‘ отримана сума
PRINT”Клієнту на рахунку при заданих умовах небхідно мати:”;PV;”грн”
END
```

3.3 Розгалужені програми

Розгалужені програми організовуються за допомогою умовного оператора ***IF***.

Розрізняють однорядкові і багаторядкові умовні оператори .

Формати однорядкового оператора ***IF***:

- ✓ простий однорядковий оператор

IF < умова> THEN < оператори >

При виконанні оператора ***IF*** перевіряється умова і, якщо вона істинна, то виконується дія, вказана після оператора ***THEN*** . Якщо вираз помилковий, то керування передається на оператор, наступний за оператором ***IF***.

IF x < b THEN GOTO m1
*IF x < a THEN y = SIN (x)*EXP(2 ^ LOG (x)) : GOTO m2*

- ✓ простий розширений оператор ***IF***

IF < умова> THEN < оператори 1 > ELSE < оператори 2 >

При виконанні оператора ***IF***, якщо умова істинна, то виконуються оператори, зазначені після оператора ***THEN***, в іншому випадку виконуються оператори, наступні за оператором ***ELSE***. Після виконання відповідної групи операторів керування передається на оператор, наступний за оператором ***IF***.

*IF x<a THEN y=SIN (x)*EXP(2 ^ LOG (x)) ELSE y=3 * x ^ 2-5 * x+4*

Після операторів ***THEN*** і ***ELSE*** може бути зазначено декілька операторів, розділених двокрапкою. Однак число операторів обмежено довжиною рядка. Цих недоліків позбавлений багаторядковий оператор ***IF***.

Формати багаторядкового оператора ***IF***:

Простий

IF <умова> THEN
[блок_операторів-1]

ELSE
[блок операторів-2]
END IF

Розширений

IF <умова1> THEN
[блок_операторів-1]

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

При записі операторів слід звертати увагу на структуру запису. Структура має відповідати тій, що вказана у прикладі. Перевага багаторядкового оператора ***IF*** полягає в тому, що кількість операторів у групах необмежена.

Як приклад програми розгалуженого обчислювального процесу програми розглянемо задачі:

1 Розрахувати, що вигідніше для вкладника: отримати 20000 грн сьогодні або отримати 35000 грн через 3 роки, якщо річна відсоткова ставка дорівнює 17 %.

$$PV = \frac{FV}{(1+\frac{pr}{100})^t},$$

де PV − поточна сума;

FV − майбутня сума;

pr − ставка дисконтування;

t − кількість років.

Розрахуємо майбутню вартість 20 000 грн через 3 роки, під 17 % річних за формулою $FV = PV*(1+\frac{pr}{100})^t$. Порівняємо отриману суму із заданою.

```
CLS                          ' очищення екрана виводу
t=3                          ‘ термін вкладу
pr=14                        ’ річна ставка
PV=20000                          ‘поточна сума
FV=PV*(1+pr/100)^ t               ‘ отримана сума
IF FV<35000 THEN
    PRINT ”Отримати 35000 грн через 3 роки є більш вигідним
    рішенням при даному значенні відсоткової ставки.”
ELSE
    PRINT ” Отримати 35000 грн. через 3 роки є менш
    вигідним рішенням при даному значенні відсоткової
    ставки.”
ENDIF
END
```

2 Торговий агент отримує відсоток від суми угоди. Якщо обсяг угоди до 3000, то 5 %; якщо обсяг до 10 000, то 2 %; якщо вище 10 000, то 1,5 %. Обчислити розмір винагороди.

SUM − сума зробленої угоди;
pr − річний відсоток угоди;
VZ − розмір винагороди.

```
CLS                                  ' очищення екрана виводу
REM Обчислення суми винагороди
INPUT „Сума угоди-”, SUM             ‘ введення суми угоди
IF SUM <3000 THEN
      pr= 5
ELSEIF SUM <10000 THEN
      pr= 2
ELSE
      pr=1.5
ENDIF
VZ=SUM*pr/100
PRINT ”Сума винагороди :”;VZ;”грн”
END
```

3.4 Циклічні програми

Циклом називається процедура, в якій обчислювальні операції виконуються багаторазово задану кількість разів або до досягнення деякої змінної, що обчислюється в тілі циклу, певного, наперед заданого значення.

Цикли можуть бути організовані з використанням операторів: ***FOR*/*NEXT***, ***WHILE* / *WEND***, ***DO* / *LOOP*.**

3.4.1 Оператор FOR / NEXT

Оператор *FOR* служить для організації циклів із заданим числом повторень. Формат оператора:

FOR X=Хпоч TO Хкін STEP Нх
< тіло циклу >
NEXT X

У даному форматі *Хпоч* − початкове значення параметра циклу, *Хкін* − кінцеве значення параметра циклу, а Нх − крок збільшення значення параметра циклу.

Оператори *FOR* і *NEXT* утворюють операторні дужки, між якими укладено тіло циклу.

3.4.2 Оператор WHILE / WEND

Оператор *WHILE* застосовується для організації циклів, кількість повторень якого не визначено заздалегідь. Умовою закінчення циклу є досягнення функцією, що обчислюється в тілі циклу, заданого значення. Формат оператора:

WHILE < умова>
< тіло циклу >
WEND

Тіло циклу виконується в тому випадку, якщо умова істина. Якщо умова хибна, то програма виходить з циклу. Особливістю використання даного оператора є те, що до початку циклу необхідно визначити результат умови.

3.4.3 Оператор DO / LOOP

Оператор *DO* − універсальний оператор, служить для організації циклів типу " ПОКИ ". Він може бути організований як цикл з передумовою, так і як цикл з постумовою. Коли в тілі циклу використовується оператор *WHILE*, то тіло циклу виконується в тому випадку, якщо умова істинна. Коли використовується оператор *UNTIL*, то тіло циклу виконується в тому випадку, якщо умова помилкова.

Формат оператора:

а) цикл з передумовою	**б) цикл з постумовою**
DO <WHILE|UNTIL>	***DO***
< тіло циклу >	**< тіло циклу >**
LOOP	***LOOP <WHILE|UNTIL>***

Як приклад програми циклічного обчислювального процесу програми розглянемо задачу.

У фінансову установу надійшла заявка від клієнта з проханням прийняти у нього грошові кошти на депозитний рахунок за такими умовами:

- сума депозиту − 10 000грн;
- термін − на 5 років;
- відсоткова ставка − 16 % річних.

Відсотки нараховуються за складною схемою.

Розрахувати суму, яку отримає клієнт після закінчення терміну договору.

SUM – накопичена сума депозиту;
pr – річний відсоток угоди;
sr – відсоткова ставка за півроку;
x – складна формула нарахувань.

```
CLS                              ' очищення екрана виводу
REM Обчислення суми депозиту
SUM=10000                        ‘початкова сума на депозитному
рахунку
t=5                              ‘термін вкладу
pr=16                            ’річна ставка
sr=pr/2                          ’піврічна ставка
FOR   i=1   TO 2*t   STEP1       ’початок циклу
      X=SUM*PR/100               ‘складна формула нарахувань
      SUM=SUM+X                  ‘формула накопичення суми
депозиту
NEXT                             ’завершення циклу
PRINT ” Сума, яку отримає клієнт після закінчення терміну
договору:”;SUM;”грн.”
END
```

СПИСОК ЛІТЕРАТУРИ

1 Быков В.Л. Основы информатики: Конспект лекций. – Брест: БГТУ, 2003.

2 Михайлов В.Ю., Степанников В.М. Современный бейсик для IBM PC. – М.: Изд-во МАИ, 1993. – 288 с.

3 Інформатика, комп’ютерна техніка, комп’ютерні технології: Посібник / За ред. О.І. Пушкаря. – К.: Академія, 2001. – 696 с.

4 Філіппенко І.Г., Гончаров В.О., Меркулов В.С. Програмування інженерно-технічних задач в середовищі QBASIC: Конспект лекцій з курсу "Обчислювальна техніка та програмування”.−Харків: УкрДАЗТ, 2007. – Ч. 3. − 134 с.