Московский государственный технический
университет гражданской авиации

Т.И. Андреева, Кишенский С.Ж.

ПОСОБИЕ
к лабораторным работам
по дисциплине
“ИНФОРМАТИКА”

часть2

для студентов I курса

специальностей

160901, 160905, 160903,280102

дневного обучения

МОСКВА 2009г.

Рецензент Мухамед Аль-Натор
Андреева Т.И., Кишенский С.Ж.
Пособие к лабораторным работам по дисциплине “ИНФОРМАТИКА”, часть 2 – М.:МГТУ ГА, 2008.- 64с.

Данное пособие издается в соответствии с учебным планом для студентов первого курса специальностей 160901, 160905, 160903,280102 дневного обучения.

Рассмотрено и одобрено на заседании кафедры №2 от 28 10.2008г. и методического совета по спец.230401 от 28.10. 2008г.

Введение

Данные методические указания предназначены для студентов 1-го курса дневной формы обучения специальностей 160901, 160905, 160903,280102 и содержат описания 6 лабораторных работ. Предполагается выполнение предложенных лабораторных работ во втором семестре после сдачи студентами первой части лабораторного практикума и приобретения ими навыков работы на компьютерах IBM PC.

Для успешного выполнения лабораторных работ данного сборника студенты должны уметь работать с операционной системой Windows, знать назначение и возможности применения основных команд MS DOS, освоить приемы работы с операционной оболочкой Norton Commander, приобрести навыки работы в текстовых редакторах.

Выбор в качестве языка программирования алгоритмического языка БЭЙСИК объясняется тем, что он является одним из самых простых и доступных для широкого круга пользователей ПЭВМ. Предлагаемая версия БЭЙСИКа (QB45,QuickBASIC, QB)- это развитый структурный и процедурный язык программирования.

Целью выполнения лабораторных работ сборника является приобретение практических навыков программирования различных типов алгоритмов, отладки программ на ПЭВМ и освоение системы программирования.

Интегрированная среда рассматриваемой системы QB45 имеет мощный экранный текстовый редактор, управляющую среду с многооконными меню, подсистему помощи, отладчик и встроенный компилятор.

Предполагается, что студенты приходят на занятия, предварительно изучив методические указания и подготовив программы индивидуальных заданий.

По каждой лабораторная работе подготавливается отчет, и работа защищается студентом.

Отчет должен содержать:

- конспект, в котором в краткой форме отражаются все разделы лабораторных работ;
- схемы алгоритмов решения заданий
- программы, реализующие алгоритмы;
- выводы по работе.

Работа в интегрированной среде Qbasic (версия qb45)

НАЧАЛЬНАЯ ИНФОРМАЦИЯ О СИСТЕМЕ QUICKBASIC

Система программирования QuickBASIC - интегрированная система, включающая текстовый редактор, управляющую среду с многооконными меню, подсистему помощи HELP, отладчик и встроенный компилятор. Система создавалась для работы в операционной системе MSDOS.

ЗАГРУЗКА СИСТЕМЫ QUICKBASIC

Загрузка (вход в систему) QuickBASIC может осуществляться следующими способами:

- двойным щелчком по ярлыку на рабочем столе **Windows.**
- через программы **Проводник, Мой компьютер** или **NC**: после запуска одной из этих программ открыть папку ***QBasic (QB,QB45)***, установить указатель мыши на файле ***Qbasic.exe (QB.exe)*** и дважды щелкнуть или нажать **Enter**;

После входа в QuickBASIC на экране появляется окно редактора, в центре которого на сером фоне может стоять приглашение в систему, в этом случае нажмите **Esc**. Экран примет вид, представленный на рисунке 1.

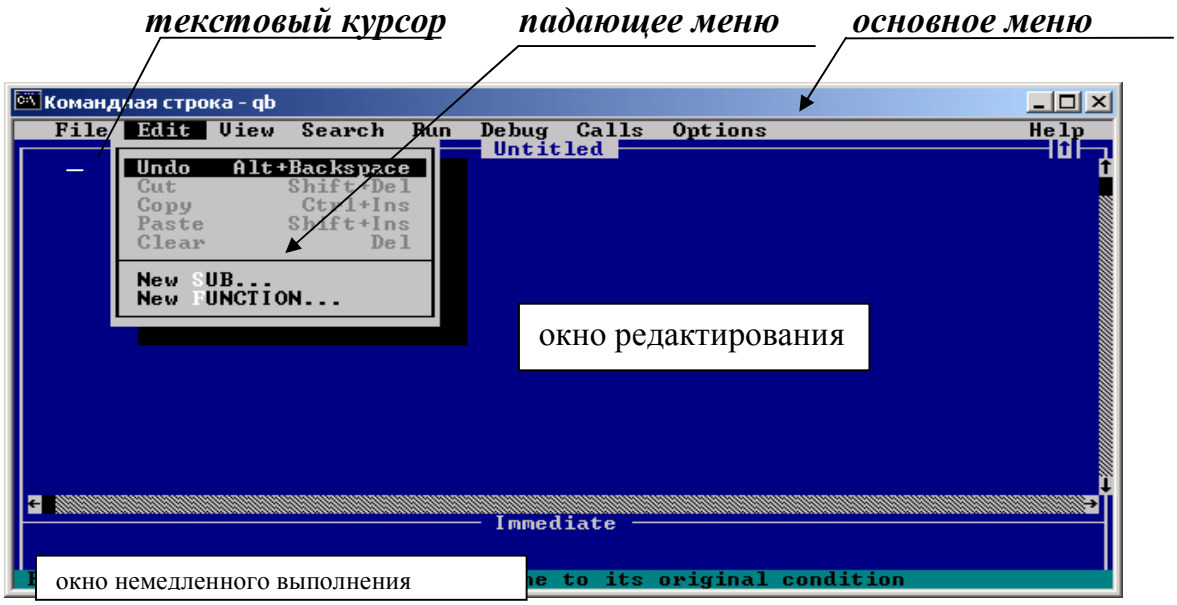


Рисунок 1 - Экран редактора QuickBASIC, версия 4.5

На экране можно выделить три основные зоны:

- верхняя строка экрана, основное меню системы;
- окно редактирования с курсором в начале первой строки окна;
- нижнее прямоугольное окно с названием Immediate (немедленно).

Основное меню системы QuickBASIC, состоит из ряда пунктов:

File - работа с файлами и операционной системой;
Edit - ввод и редактирование текста;
View - отображение отдельных страниц с текстами;
Search - работа с фрагментами текста (поиск, выделение, замена);
Run - запуск программ;
Debug - отладка программ;
Calls - вызов отдельных процедур;
F1=Help - помощь.

Основное меню предназначено для выбора режимов работы системы. Вход в меню осуществляется щелчком мыши по выбранному пункту, после чего раскрывается (падает) подменю выбранного пункта. Подменю содержит список команд (Рисунок 1).

С помощью клавиатуры также можно войти в меню, нажав клавишу **Alt** , при этом выделится пункт **File**. Дальнейший выбор пункта меню осуществляется передвижением курсора с помощью клавиш управления курсором и нажатием клавиши Enter, после чего откроется подменю.

Подменю позволяет детализировать работу выбранного режима. Для выполнения команды подменю следует установить на нее указатель мыши (текстовый курсор) и щелкнуть (нажать ENTER).

Часто вход в пункт подменю открывает диалоговое окно. Диалоговые окна обеспечивают дальнейшую детализацию выбранного режима и назначения необходимых условий работы в нем (Рисунки 2, 3).

Диалоговые окна имеют несколько полей, ограниченных прямоугольной рамкой. Переход от одного поля к другому осуществляется щелчком мыши или нажатием клавиши TAB . Курсор, находящийся в окне указывает на активное поле, в котором можно осуществлять необходимые записи, (например: записать путь к файлу). Выход из окна осуществляется нажатием клавиши ESC .

Текстовый редактор системы QuickBASIC

Окно редактирования предназначено для записи и редактирования программ с использованием встроенного текстового редактора системы QuickBASIC.

При загрузке системы вышеуказанным способом, автоматически устанавливается режим редактирования. После входа в режим редактирования курсор установлен в левом верхнем углу чистого поля экрана и показывает место, с которого можно набирать программу. Текст программы вводится с клавиатуры.

Далее перечислены наиболее часто употребляемые команды текстового редактора, выполняемые с помощью клавиш управления курсором:

Смещение курсора:

- на символ влево, вправо, вверх, вниз;

PGUP - на экранную страницу вверх;

PGDN - на экранную страницу вниз;

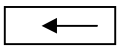
HOME - в начало строки;

END - в конец строки;

CTRL-HOME - в начало текста;

CTRL-END - в конец текста.

Команды редактирования:

 , BACKSPACE - стереть символ слева от курсора;

SHIFT - СМЕЩЕНИЕ КУРСОРА - выделение текста;

SHIFT - DEL - удаление выделенного текста с сохранением в буфере;

SHIFT - INS - восстановить удаленный текст из буфера;

DEL - стереть символ над курсором, удалить выделенный текст;

ENTER - вставить пустую строку, разрезать строку;

CTRL - Y - удалить текущую строку с сохранением в буфере;

Нормальный режим работы редактора - режим **вставки,** текстовый курсор имеет вид подстрочной черточки. В режиме вставки пропущенный символ вставляется перед тем символом, под которым установлен курсор.

Для перехода к режиму **замены**, нажимают клавишу **INS,** текстовый курсор примет форму прямоугольника. В этом режиме ошибочный символ **заменяется на правильный**, если под ним установлен курсор.

Выполнение программы

После загрузки системы программирования необходимо:

- ввести текст программы
- отладить программу;
- выполнить и получить результат.

После ввода текста программы следует:

а) войти в основное меню;

б) установить курсор на пункт **RUN** и нажать **ENTER** ;

в) в открывшемся подменю выбрать пункт **START** и нажать **ENTER**.

Пункты а-в) можно заменить нажатием "горячих клавиш" - **SHIFT-F5**.

После запуска программы на выполнение возможны две ситуации:

- в программе транслятор системы обнаружил ошибки;
- в программе ошибок не обнаружено.

Если в программе имеются ошибки, то сообщение о первой высвечивается в открывающемся на экране окне, а место ошибки отмечается в тексте программы курсором. Для продолжения работы нужно нажать клавишу **Esc**, исправить ошибку и снова запустить программу на выполнение.

Если больше ошибок не обнаружено, то программа выполняется и на экране появляется результат. В нижней строке экрана сообщение "*Press any key to continue*". Нажмите любую клавишу для продолжения, и Вы вернетесь в окно редактора. Для повторного просмотра результатов нужно нажать **F4.**

Создание и Сохранение программы - команды меню File

В таблице приведены основные команды подменю File:

Команда подменю	Выполняемое действие
New Program	- устанавливается режим редактирования. При выборе этого режима очищается ОЗУ от старых программ и подготовка к вводу новой программы под именем UNTITLED, которое при сохранении файла на диске нужно изменить.
Open Program	- загружается ранее созданный файл;
Save , Save As	- команды позволяют сохранить программу на диске;
Dos Shell	- временный выход в среду MS-DOS, возврат в QBASIC осуществляется командой **EXIT;**
EXIT	- выход из среды QBASIC

При выполнении программы, набранной непосредственно в редакторе, вся информация хранится в оперативной памяти ЭВМ. Чтобы сохранить программу на диске, необходимо установить курсор на пункт верхнего меню **File** и нажать **Enter.** Откроется подменю, в котором нужно выбрать пункт **Save**. Система откроет для этого новое диалоговое окно (Рисунок 2) и предложит вам ввести имя файла программы в поле окна **File name**. В поле **Dirs/Drives** выберите диск и каталоги, а в поле **Format – Text**, затем нажмите **Enter**.

При повторном сохранении программы ее имя не запрашивается

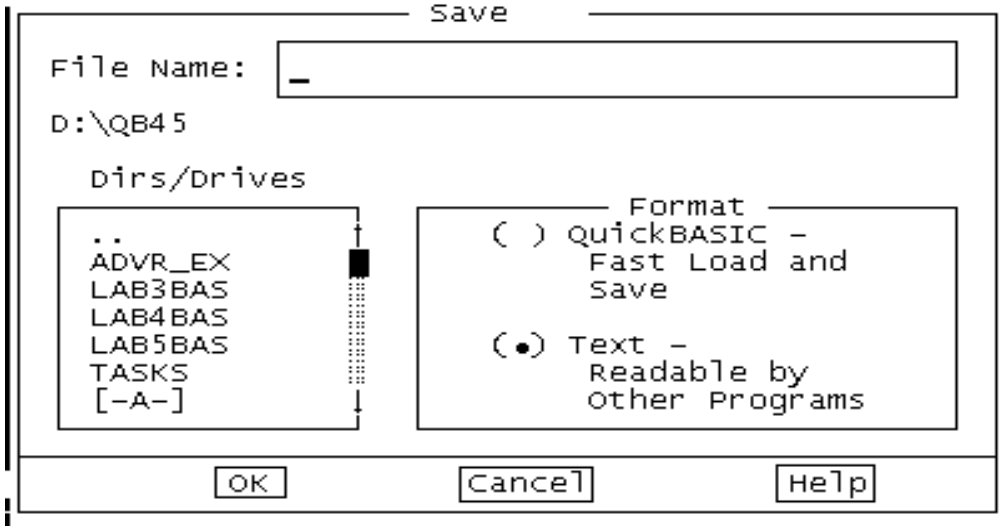


Рисунок 2. Диалоговое окно Save

В пункте меню **File**, также имеется опция **Save As** (сохранить как), которая позволяет сделать копии файлов под новыми именами, сохранив при этом оригинал, или записать на диск измененный файл под новым именем или в другой папке. При обращении к этой опции система открывает диалоговое окно, подобное Save.

Сохраняйте программу по частям в процессе ее набора

При обращении к сохраненному ранее файлу нужно войти в меню **File** и выбрать пункт **Open Program.** Откроется диалоговое окно, представленное на рисунке 3. С помощью мыши в поле **Dirs/Drives** следует выбрать диск, а затем папку. В поле **Files** выбрать файл.

Для ввода следующей программы нужно войти в меню **File** и выбрать пункт **New Program.** Окно очистится, и Вы приступаете к набору текста следующей задачи.

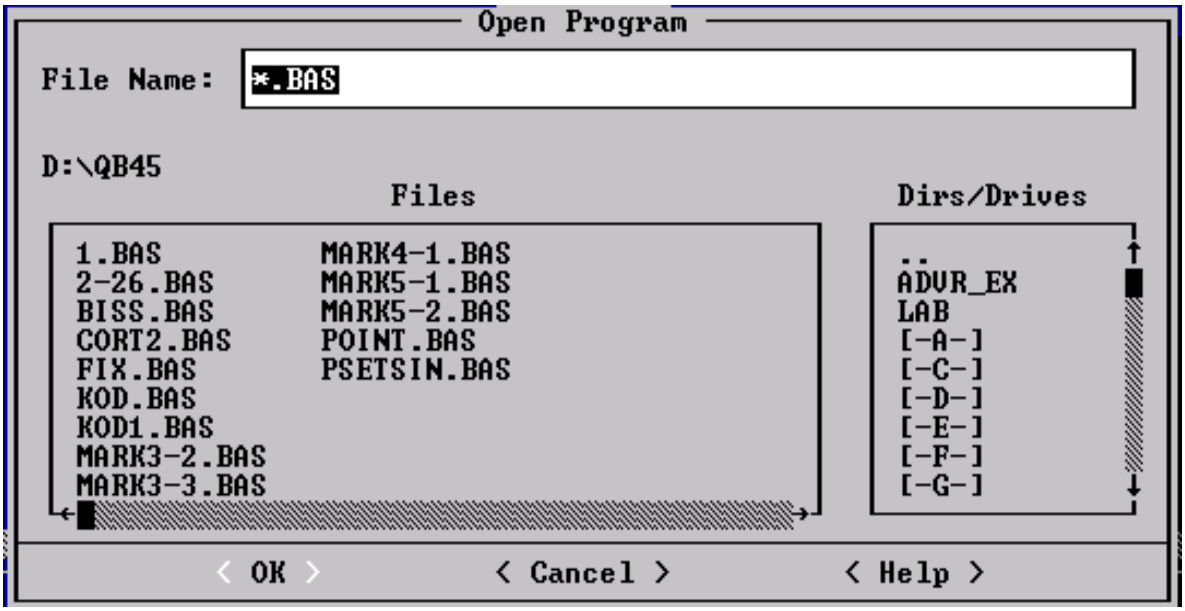


Рисунок 3 Диалоговое окно " Open Program"

ЗАПУСК И ПЕРЕЗАПУСК ПРОГРАММЫ - КОМАНДЫ МЕНЮ RUN

Ранее уже говорилось о том, как программа запускается на выполнение с помощью подменю Run. Но этот пункт основного меню среды предоставляет набор команд, позволяющих управлять программой, режимами компилятора и редактора связей. После входа в пункт Run открывается подменю:

Команда подменю	Выполняемое действие
Start, Shift+F5	-запускает программу на выполнение
Restart	- устанавливает пошаговый режим исполнения программы, переход к следующему шагу осуществляется нажатием F8. Этот режим удобен при отладке программы совместно с использованием режима отображения дисплейной памяти, который устанавливается клавишей F4;
Continue, F5	- продолжение выполнения программы с точки ее останова;
Make EXE File	- позволяет создать исполняемый файл с расширением "exe". Такой файл запускается из ОС.

ОТЛАДКА ПРОГРАММЫ - КОМАНДЫ И РЕЖИМЫ МЕНЮ DEBUG

Пункт **Debug** основного меню среды задает режимы выполнения программы и позволяет отображать промежуточные результаты на этапе отладки. После входа в пункт **Debug** открывается подменю, список команд которого приведен в таблице:

Команда подменю	Выполняемое действие
Add Watch	- позволяет указать имена переменных и выражения, значения которых будут отображаться в окне под основным меню. Окно удобно использовать при пошаговом исполнении программы;
Watchpoint	- позволяет указать имена переменных и выражения **логического** типа, значения которых проверяются на достижение значения **True** (истина). Информация отображается в окне под основным меню. Как только условие выполнено, программа приостанавливается;
Delete Watch	- используется для удаления из окна отдельных переменных или выражений;
Delete All Watch	- полностью удаляет окно со всеми контролируемыми переменными;
Trace On	- включает и выключает режим трассировки.
Toggle Breakpoint, F9	- включает или выключает режим прерывания программы в тех строках, где находится курсор;
Clear All Breakpoint	- отключает все ранее установленные прерывания.

ПОДСИСТЕМА ПОМОЩИ - HELP

Вход в подсистему осуществляется выбором пункта меню HELP основного меню. В развернувшемся подменю предлагается четыре режима помощи, список которых приведен в таблице:

Режим помощи	Выполняемое действие
Index	Выдается справка по всем ключевым словам, операторам и функциям.
Contents	Выдается перечень разделов справочника, по которым пользователь может получить справку.
Topic	Справка по конкретному оператору или функции (Shift-F1) .
Help	выдается справка о самой подсистеме помощи (F1)

ИСПОЛЬЗОВАНИЕ ОКНА "IMMEDIATE "

При работе в среде QuickBASIC, возможны два способа исполнения программных строк: автоматический, т.е. в соответствии с введенной программой, и командный. В первом случае осуществляется компиляция программы в памяти, а затем производится исполнение.

В командном режиме работы возможно непосредственное исполнение отдельных программных строк. Этот режим работает, если программные строки или операторы занесены в окно "Immediate " и запускаются на выполнение нажатием клавиши **Enter**. Для того чтобы поместить текст строки в окно **"Immediate "** нужно нажать клавишу **F6**, курсор переместится в окно и затем набрать нужный оператор. Также можно поместить текст оператора в окно следующим образом:

отметить текст - SHIFT + клавиши управления курсором;
скопировать текст в буфер - CTRL + INS;
перейти в окно "Immediate " - F6;
скопировать текст из буфера - SHIFT + INS.

ЛАБОРАТОРНАЯ РАБОТА N6
ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Цель работы:

1. Приобретение навыков в составлении простейших программ на алгоритмическом языке Бэйсик.
2. Изучение среды программирования QuickBASIC.
3. Приобретение навыков работы в редакторе QuickBASIC.

ПРОГРАММЫ НА ЯЗЫКЕ БЭЙСИК

Решение задачи с помощью ЭВМ состоит из следующих этапов:

- математической формулировки условия задачи;
- выбора численного метода ее решения;
- разработка алгоритма;
- составление программы на алгоритмическом языке;
- отладка программы.

В предлагаемых заданиях к лабораторным работам условия задач уже представлены в математической формулировке с указанием численного метода решения и необходимость в выполнении первых двух этапов отпадает.

Программа на языке Бэйсик - это последовательность строк, описывающих алгоритм решения задачи. Строка может содержать один или несколько операторов, разделенных двоеточием, а также комментарии.

Оператор представляет собой строго формализованное указание на выполнение конкретного действия.

Каждая строка может начинаться метки. Метка может быть цифровой или буквенно-цифровой. Буквенно-цифровые метки могут иметь от 1 до 40 символов и начинаться с буквы, а завершаться двоеточием, тогда как цифровая метка завершается пробелом. Метка не определяет порядок выполнения строк программы, не обязательна, и служит, как правило, для ссылки на нее. Программные строки выполняются в порядке их записи. Длина программной строки не должна превышать 256 символов.

Современные алгоритмические языки используют наборы различных типов данных.

Программа, написанная на языке Бэйсик, обрабатывает числовые и символьные данные. Данные представляются в программе в виде констант и переменных. Тип данных определяет возможные значения констант и переменных, форму представления в ЭВМ, объем занимаемой памяти, операции, которые могут выполняться над данными этого типа.

Числа. Бэйсик оперирует двумя типами чисел: вещественными и целыми.

Под целое число отводится 2 байта памяти, и оно хранится форме с фиксированной точкой. Запись целого числа представляет собой последовательность цифр со знаком или без него (например:5487, -7821, +3841).

Вещественные числа хранятся в ячейке памяти длиной 4 байта в форме с плавающей точкой. Возможны две формы " внешней " записи вещественных чисел в программах:

- с фиксированной точкой (например, - 3.7);
- с плавающей точкой (например: -00.45E2, 0.78D-3, здесь буквы " E " и " D " означают основание " 10 ", обычной и двойной точности соответственно, разделяют мантиссу и порядок).

Числовое или символьное значение может быть присвоено переменной или константе.

Переменная - величина, которая может меняться при выполнении программы. Переменная всегда имеет имя, которое содержит не более 40 буквенно-цифровых символов и начинается с латинской буквы.

Способы описания типа данных в Бэйсике:

1. Явно - с помощью определенных суффиксов, которые добавляются к имени переменных или констант.
2. Явно - с помощью операторов описания типа.
3. Неявно - с помощью оператора объявления типа данных по первой букве имени переменной.

Таблица диапазона числовых данных

тип	диапазон
целый	-32768 ÷ +32767
длинный целый	-2147483648 ÷ +2147483647
веществ. обычной точности	-3.402823E+38 ÷ -1.40129E-45 +1.40129E-45 ÷ +3.402823E+38
веществ. двойной точности	-1.79769E+308 ÷ -4.94965E-324 +-4.94965E-324 ÷ +1.79769E+308

Таблица описания типа данных:

тип	суффикс (явно)	оператор описания (явно)	оператор объявления (неявно)	объем памяти в байтах
целый	%	DIM имя as integer	DEFINT	2
пример	NAME1%	DIM NAME1 AS INTEGER	DEFINT N	
длинный целый	&	DIM имя as LONG	DEFLNG	4
пример	NAME2&	DIM NAME2 AS LONG	DEFLNG N	
веществ. обычной точности	!	DIM имя as SINGLE	DEFSNG	4
пример	NAME3!	DIM NAME3 AS SINGLE	DEFSNG N	
веществ. двойной точности	#	DIM имя as DOUBLE	DEFDBL	8
пример	NAME4#	DIM NAME4 AS DOUBLE	DEFDBL N	
символьный	$	DIM имя as STRING	DEFSTR	4+nбайт (n-кол-во символов)
пример	NAME5$	DIM NAME5 AS STRING	DEFSNG N	

Константы. Значения констант не меняются в процессе работы программ. В Бэйсике различают два вида констант: - неименованные и именованные. Константы бывают числовые и символьные. Неименованная числовая константа - это число, а именованная константа должна быть объявлена с помощью ключевого слова CONST, например:

```
CONST   PI = 3.14
CONST   PL = 0.23E-3        ' ФОРМА Е
CONST    Z$ = " ПРИВЕТ " ' СИМВОЛЬНАЯ
```

Тип числовой переменной или константы можно не указывать в программе, тогда автоматически он становится - SINGLE.

В данной работе мы ограничимся рассмотрением числовых данных.

ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Алгоритм - это последовательность действий, однозначно определяющих процесс преобразования исходных и промежуточных данных в результат решения задачи. Форма представления алгоритма может быть как текстовой, так и графической - в виде схемы. Решение всего многообразия задач может быть сведено к трем типам алгоритмов: линейному, разветвляющемуся и циклическому. Чаще встречается комбинация этих типов.

Линейный алгоритм - алгоритм, в котором к результату решения задачи приводит последовательное выполнение действий.

Алгоритм решения такой задачи в словесной форме состоит из следующих пунктов: начало программы; ввод исходных данных; вычисления; вывод результатов; окончание программы.

Программы, реализующие линейный алгоритм, как правило, очень просты и для их реализации используются операторы ввода данных, выполнения действий (вычислений) и вывода результатов.

ВВОД ДАННЫХ

Ввод данных в программах, написанных на Бэйсике можно осуществлять несколькими способами:

1) **input a, b, c - оператор ввода данных в диалоговом режиме, где a, b, c - список имен переменных**

Встретив этот оператор, машина останавливает выполнение программы. На экране появляется знак "**?**", затем значения переменных нужно ввести через запятую и нажать **Enter**. Количество, последовательность и тип вводимых данных должны соответствовать именам переменных оператора "**input**".

В списке ввода оператора "**input**" на первом месте можно дать подсказку - текст, заключенный в двойные кавычки, например: **input "a,b,c:",a,b,c**

При выполнении такого оператора "**input**" на экране появится подсказка (**a,b,c:**), после чего нужно ввести значения переменных и нажать **Enter**.

2) **read a, b, c; data** - оператор **input** считывает данные, перечисленные в операторе **data,** причем количество данных должно соответствовать переменным в операторе **read** по количеству, типу и порядку следования, например: **data 5, -8, 12.5**
read a, b, c

При выполнении оператора **read a, b, c:**

- переменной **a** будет присвоено значение 5
- переменной **b** будет присвоено значение -8
- переменной **c** будет присвоено значение 12.5

3) задать исходные данные можно также операторами присваивания, например: **a = 5 : b = -8 : c = 12.5**

ВЫПОЛНЕНИЕ ВЫЧИСЛЕНИЙ

Для вычисления арифметических выражений используется оператор **присваивания**, частный случай которого применяется и для ввода данных.
Общий вид оператора: **Q=P ,**
где **Q** - имя переменной;
P - арифметическое выражение

Арифметические выражения соответствуют общепринятым алгебраическим выражениям, в них могут входить числа, переменные, функции, соединенные знаками арифметических операций и круглыми скобками.

В Бэйсике имеются следующие арифметические операции:

^ - возведение в степень;
*,/ - умножение, деление;
\ - целочисленное деление;
MOD - определение остатка от деления;
+,- - сложение, вычитание.

Операции перечислены в порядке убывания приоритета их выполнения. Действия внутри круглых скобок выполняются первыми.

Примеры записи некоторых арифметических операций и их результаты приведены в таблице.

ВЫРАЖЕНИЕ	РЕЗУЛЬТАТ	ПРИМЕЧАНИЕ
2 ^ -2	0.25	возведение в степень -2
7 / 2	3.5	
7 \ 2 1.6 \ 2.2	3 1	делимое и делитель (операнды) округляются, тип результата INTEGER
7 MOD 2 8.3 MOD 3.3 8.6 MOD 3.3	1 2 0	операнды округляются, тип результата INTEGER

Математические функции. При записи функций на Бэйсике *аргумент функции заключается в круглые скобки*. В качестве аргумента математических функций может быть число, переменная или арифметическое выражение. Следует заметить, что в тригонометрических функциях аргумент должен быть задан в радианной мере.

Наиболее распространенные функции языка Бэйсик:

ABS(X) - вычисляет модуль аргумента, что соответствует математической записи $|x|$;

EXP(X) - экспонента, соответствует математической записи e^x ;

LOG(X) - вычисляет натуральный логарифм аргумента, что соответствует математической записи **ln(x)**;

SQR(X) - вычисляет корень квадратный из аргумента;

ATN(X) - вычисляет арктангенс аргумента;

COS(X) - вычисляет косинус аргумента;

SIN(X) - вычисляет синус аргумента;

TAN(X) - вычисляет тангенс аргумента;

RND(X) - выдает случайное число обычной точности в интервале 0÷1. Аргумент может быть опущен. Рекомендуется в начале программы запустить генератор случайных чисел оператором **RANDOMIZE TIMER**.

SGN(X) - определяет знак аргумента. Если аргумент отрицательный, функция принимает значение (**-1**,) если положительное (**+1**). При нулевом аргументе функция также принимает значение **0**

FIX(X) - отбрасывает дробную часть значения аргумента

INT(X) - округляет аргумент в сторону уменьшения

CINT(X) - округляет аргумент по математическим правилам.

Примеры записи функций округления и их результаты в таблице:

Выражение	Результат	Выражение	Результат	Выражение	Результат
FIX(5.7)	5	INT(5.7)	5	CINT(5.7)	6
FIX(5.1)	5	INT(5.1)	5	CINT(5.1)	5
FIX(-5.7)	-5	INT(-5.7)	-6	CINT(-5.7)	-6
FIX(-5.1)	-5	INT(-5.1)	-6	CINT(-5.1)	-5

Для более подробного ознакомления с набором встроенных функций и их синтаксисом необходимо обратиться к документации по описанию конкретной версии языка.

ВЫВОД ДАННЫХ И РЕЗУЛЬТАТОВ

print x,y,z - оператор вывода данных и результатов,

где **x,y,z -** список элементов вывода.

В качестве элементов вывода могут быть имена переменных, арифметические выражения, а также текст, заключенный в двойные кавычки.

При выполнении оператора на экран выводятся значения переменных, арифметических выражений, текст. Список может отсутствовать и в этом случае на экране пропускается строка.

Разделителем элементов вывода может быть запятая или точка с запятой, от этого зависит интервал в строке вывода на экране между выводимыми данными. В BASICe строка делится на пять равных зон. Если разделителем является запятая, то очередной элемент выводится в начале следующей зоны. В том случае, когда разделителем является точка с запятой, очередной элемент выводится через пробел.

ПРИМЕР 6.1 Составить программу вычисления и вывода на экран радиусов описанной и вписанной окружностей R1и R2 правильного многоугольника, а также площади правильного многоугольника. Количество сторон многоугольника - n и длину его стороны - a задать с экрана монитора. Для вычисления воспользуемся следующими формулами:

$R1 = \dfrac{a}{2\sin(3{,}14/n)}$ - радиус описанной окружности;

$R2 = \dfrac{a}{2tg(3{,}14/n)}$ - радиус вписанной окружности;

$S = \dfrac{n \cdot a \cdot R2}{2}$ - площадь правильного многоугольника.

Алгоритм решения задачи в словесной форме состоит из следующих пунктов: начало; ввод значений переменных **a** и **n**; вычисление функции R1, R2 и S; вывод значений функций R1, R2 и S; окончание программы.

```
   REM Пример 6.1 Вычисление по формулам
CLS
PRINT "Введите значения переменной N"
      INPUT N
      INPUT  "Введите значение переменной A"; A
      R1=A/(2*SIN(3.14/N)): R2 = A/(2*TAN(3.14/N)): S = N*A*R2/2
      PRINT "N="; N,"A="; A, "R1="; R1, "R2="; R2,"S=";S
      END
```

Пояснения к программе, которая реализует алгоритм**:**
- оператор **REM**, позволяет вводить комментарии, пояснения к программе;
- **CLS** очищает экран;
- **PRINT** предназначен для вывода на экран текста, заключенного в апострофы, и значений переменных;
- **INPUT** служит для ввода значений переменных по запросу после "**?**";
- для вычисления искомых величин используется оператор присваивания,;
- **END** окончание программы.

ПРИМЕР 6.2 Составить программу вычисления по формуле и вывода на экран результата вычислений. $z = 7.5e^{xy} + \sqrt[3]{\frac{5y}{7x}} + \log_9(xy)$

```
  REM Пример 6.2 Вычисление по формуле
CLS
    INPUT "Введите через запятую значения переменных x,y ", X,Y
    P=X*Y ' отдельно вычислим xy и обозначим      P
    Z=7.5*EXP(P)+(5*Y/(7*X))^(1/3)+LOG(P)/LOG(9    )
    PRINT "Z="; Z
    END
```

Пояснения к программе:

В приведенной программе используются те же операторы, что и в примере 6.1. Следует обратить внимание на запись арифметического выражения:

- скобки определяют последовательность выполнения вычислений, количество открытых скобок равно количеству закрытых;
- для вычисления корня использовано возведение в степень;
- для вычисления логарифма по основанию 9 используется формула перехода от одного основания к другому.

Комментарий также можно записывать после апострофа.

Индивидуальное задание по лабораторной работе состоит из четырех задач. Примеры типовых заданий по программированию линейных алгоритмов:

Задание 1

1. $4^{\ln x} + \log_3 7x^2 - \dfrac{\sin(x+y)}{\cos(x-y)}$
2. $2x^2 + \arccos\dfrac{1}{x} \cdot \log_3 2y$
3. $6y \cdot 2x^3 + 4e^{-x} + \log_3 7x^2$

Задание 2

1. По заданным коэффициентам a1, a2, b1, b2, c1, c2 найти решение системы уравнений a1 * x + b1 * y = c1; a2 * x + b2 * y = c2
2. Определить сумму квадратов цифр, входящих в заданное трехзначное число f.
3. Найти площадь кольца, внешний и внутренний радиусы которого равны соответственно R и r.

Задание 3

1. Вычислить расстояние между двумя точками на плоскости с координатами (x1, y1) и (x2, y2).
2. Три резистора с сопротивлениями R1, R2 и R3 соединены параллельно. Определить сопротивление их соединения.
3. По заданным значениям p1 = x1 + jy1 и p2 = x2 + jy2 двух комплексных чисел определить их сумму и произведение.

Задание 4

1. $$\frac{1+\sin^2(x+y)}{2+\left|x-\frac{2x}{1+\left|\sin^2(x+y)\right|}\right|}$$

2. $$\frac{x}{1+\cos\frac{x}{1+x}}+\left|\sqrt[3]{x-y^{\sin 2x}}\right|$$

3. $$\sqrt[3]{\frac{y^2+\log_2 e^x}{\cos^2 2x^{3-1}}}-\frac{ctg^4\left|(x-1)\right|}{e^{\sin x-1}-1}$$

Лабораторное задание

1. Изучить окно редактирование системы QuickBASIC
2. Поочередно набрать тексты программ Ваших заданий
3. Каждую программу сохранить в отдельном файле
4. Программы отладить и получить результаты для различных исходных данных
5. Результаты проанализировать
6. Составить краткий конспект. Защитить работу

ЛАБОРАТОРНАЯ РАБОТА №7
ПРОГРАММИРОВАНИЕ УСЛОВНЫХ АЛГОРИТМОВ

Цель работы:

1. Дальнейшее изучение приемов программирования на алгоритмическом языке Бэйсик.
2. Программирование условных алгоритмов.
3. Дальнейшее изучение среды программирования и приемов отладки программ.

ПРОГРАММИРОВАНИЕ ВЕТВЯЩИХСЯ АЛГОРИТМОВ

Алгоритм разветвляющейся структуры - алгоритм, в котором последовательность выполнения действий зависит от каких-либо условий.

В языке Бэйсик для ветвления используются следующие операторы:

1.Оператор безусловной передачи управления **GOTO N** , - где N метка строки. Этот оператор передает управление строке с меткой N.

2. Операторы условной передачи управления (приведены три типа):

а) **IF <** логическое выражение **> THEN <** операторы **>**

- при выполнении оператора IF сначала определяется результат логического выражения: ИСТИНА (TRUE) или ЛОЖЬ (FALSE). Если ИСТИНА, то управление передается операторам, следующим за словом THEN, если - ЛОЖЬ, то оператору, записанному после оператора IF.

б) **IF <**логическое выражение**>THEN<** операторы **>ELSE<**операторы**>**

- при выполнении оператора IF данной модификации, так же сначала определяется результат логического выражения. Если ИСТИНА, то управление передается операторам, следующим за словом THEN, если - ЛОЖЬ, то оператору записанному после ELSE.

в) блочный " **IF** " (записывается в нескольких строках):

IF < логическое выражение > **THEN**
< операторы >
[ELSEIF < логическое выражение > **THEN**
< операторы >
ELSE
< операторы > **]***
END IF

При выполнении блочного IF, сначала определяется результат первого логического выражения. Если ИСТИНА, то управление передается операторам, следующим за первым словом THEN, а затем к строке следующей за END IF. Если - ЛОЖЬ, то определяется результат следующего

---

* ∗ при записи структуры операторов в общем виде квадратные скобки означают необязательность элемента, заключенного в **“ [ ] ”**

логического выражения, и в случае ИСТИНЫ управление передается операторам, записанным за следующим THEN, а потом к строке идущей за END IF и т.д. Если же ни одно из условий оператора не выполняется, то выполняются операторы, записанные после слова ELSE, а потом к строке следующей за END IF.

Логические выражения состоят из числовых или текстовых данных, знаков отношений и логических операций.

ЗНАКИ СРАВНЕНИЯ	
Название знака	В программе
Равно	=
не равно	<>
больше	>
больше или равно	>=
меньше	<
меньше или равно	<=

ЛОГИЧЕСКИЕ ОПЕРАЦИИ	
Название операции	В программе
Логическое умножение	AND
Логическое сложение	**OR**
Отрицание	**NOT**

Пример 7.1 Из двух случайных чисел X,Y вывести наибольшее и если X>Y также вывести разность этих чисел.

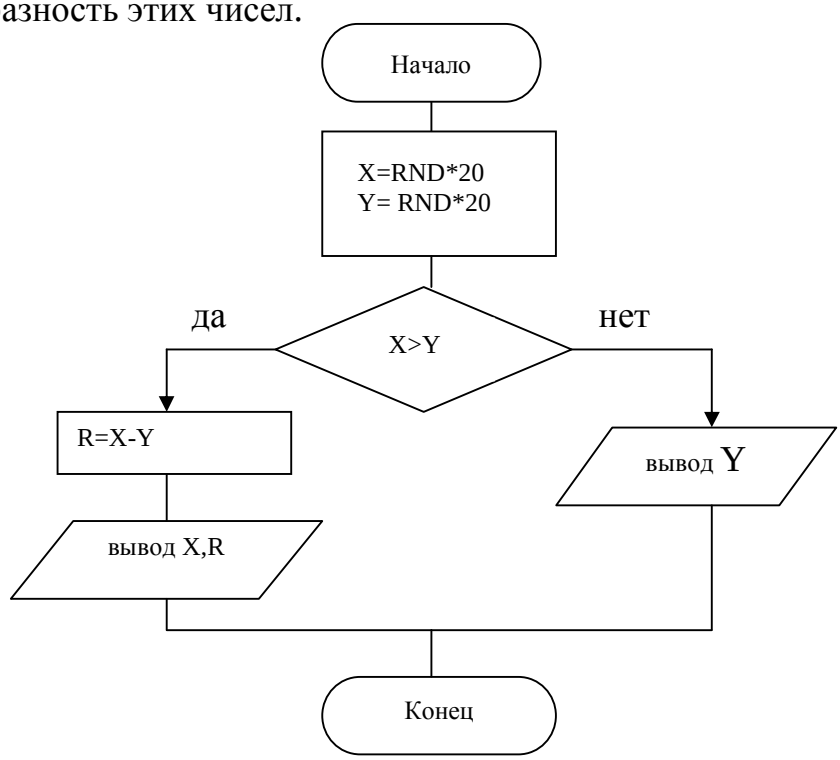


Рисунок 4. Схема алгоритма к примеру 7.1

```
REM Пример 7.1  Программирование ветвящегося алгоритма
CLS
RANDOMIZE TIMER              ' Запущен датчик случайных чисел
X=RND*20: Y=RND*20           ' X, Y - случайные числа
IF X>Y THEN R=X-Y: PRINT "X=";X,"X-Y:"; R  ELSE  PRINT "Y=";Y
END
```

Пояснения к программе:

В данной программе использован оператор IF - строчная форма. При выполнении условия X>Y после THEN в соответствии со схемой алгоритма записаны два оператора, разделенные двоеточием - оператор присваивания R=X-Y (разность чисел X и Y) и вывода результатов. Если условие не выполняется, то выводится Y после ELSE.

Пример 7.2 Составить схему алгоритма и программу вычисления и печати функции F(x) для заданного значения x:

$$F(x) = \begin{cases} \sin x & \text{, если } x \le a \\ \cos x & \text{, если } a < x < b \\ tgx & \text{, если } x \ge b \end{cases}$$

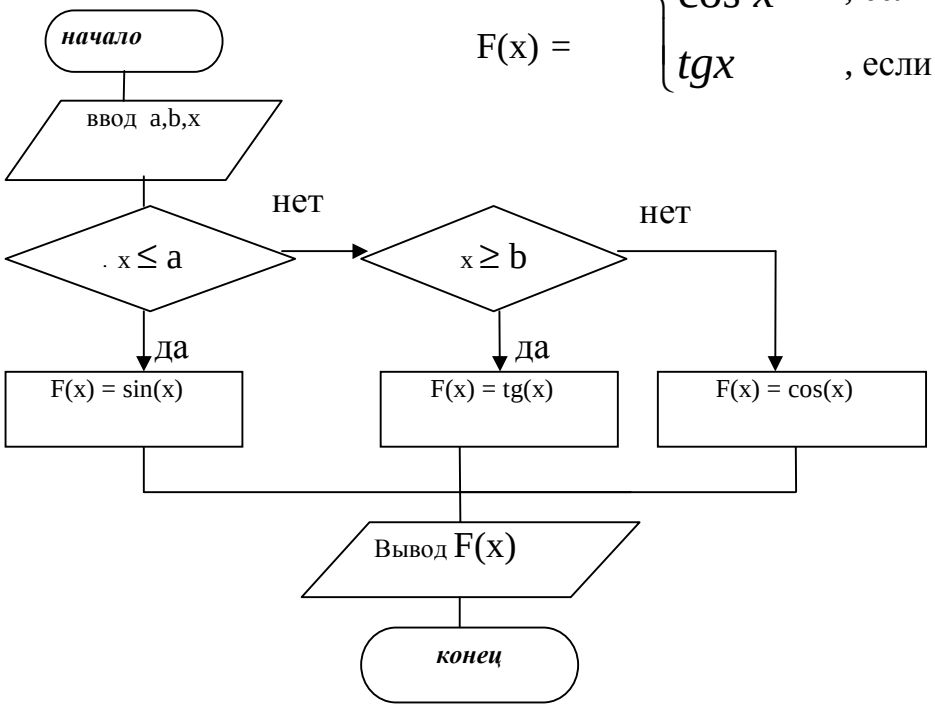


Рисунок 5 Схема алгоритма к примеру 7.2

```
REM Пример 7.2 Программирование ветвящегося алгоритма
CLS
INPUT " Введите значения переменных A,B,X   через запятые" , A,B,X
IF X <= A THEN
F = SIN (X)
ELSEIF  X >= B THEN
F = TAN (X)
ELSE
F = COS (X)
END IF
PRINT "X ="; X,"F ="; F
END
```

Пояснения к программе:

В программе для организации ввода данных в диалоговом режиме используется только оператор " INPUT " (сравните с примером 6.1)

" IF " - блочный обеспечивает ветвление. В зависимости от введенных значений переменных A, B, X процесс вычисления F пойдет в соответствии с алгоритмом (рисунок 5) по одной из ветвей.

Индивидуальное задание по лабораторной работе состоит из трех задач.

Далее приведены примеры типовых заданий по программированию условных алгоритмов.

Задание 1

1. $$F(x,y) = \begin{cases} \max(x,y+5), & \text{если } x > y \\ \min(x+1,y,3), & \text{если } x < y \\ x\ y, & \text{если } x = y \end{cases}$$

2. $$F(x,y) = \begin{cases} x + \sin(y), & \text{если } x - y > 0 \\ y - \cos(x), & \text{если } x - y < 0 \\ tg(x) + 4y, & \text{если } x = y \end{cases}$$

3. $$F(x,y) = \begin{cases} 1, & \text{если } x < 1 \\ x + 5, & \text{если } 1 < x < 2 \\ 6y \cdot 2x + 4, & \text{если } 2 < x < 4 \\ x + \sin(y), & \text{если } 4 < x \end{cases}$$

Задание 2

1. Определить, равна ли сумма первых двух цифр четырехразрядного числа **a** сумме его последних цифр.

2. Определить, делится ли заданное натуральное число **a** на 9.

3. Задан квадрат со стороной b и центром в начале координат, а также — точка с координатами x, y. Выяснить, принадлежит ли точка квадрату и одновременно находится ли вне вписанного в него круга.

Задание 3

1. Дано натуральное число a. Определить, является оно полным квадратом или кубом.

2. Имеет ли уравнение $\mathrm{arctg}(2^x - p) = 2^{1/2}$ корень на отрезке [a,b], для заданных чисел p, a, b (a < b).

3. Даны координаты (в виде двух пар чисел от 1 до 8) двух полей шахматной доски. Определить, может ли шахматный конь одним ходом перейти с одного поля на другое?

Лабораторное задание

1. Набрать, отладить и выполнить программы, реализующие условные алгоритмы Вашего индивидуального задания.

2. Создать исполняемые файлы (с расширением exe).

3. Проанализировать работу операторов, пользуясь отладочными режимами.

4. Составить краткий конспект. Защитить работу.

ЛАБОРАТОРНАЯ РАБОТА №8.
ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ.

Цель работы:

1. Изучение приемов программирования циклических алгоритмов.
2. Программирование циклических алгоритмов на языке БЭЙСИК.
3. Отладка циклических программ в среде программирования БЭЙСИК с использованием режимов "Debug".

ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ

Алгоритм называется циклическим, если все или отдельные его этапы в процессе решения задачи неоднократно повторяются.

Цикл обеспечивает повторное выполнение, или, иначе говоря, циклическую работу операторов. Оператор или группа операторов, повторяющаяся в цикле, называется "телом цикла".

Далее рассмотрим два типа циклических задач:

а) задачи, в которых вычисления многократно ведутся по одним и тем же формулам с различными значениями входящих в нее величин. Такие задачи иногда называются задачами на табулирование.

б) задачи, где значение некоторой величины вычисляется через значение этой же величины, полученное в предыдущем цикле (рекурсии). Примерами таких задач являются задачи вычисления сумм и произведений рядов, а также вычисление значений факториала.

Характерные моменты циклического алгоритма:

- первоначальный вход в цикл выполняется через блок подготовки;
- цикл всегда характеризуется некоторой переменной, называемой параметром цикла. Начальное значение параметра задается перед циклом в блоке подготовки, а при каждом повторении цикла выполняются операторы тела цикла и параметр изменяется на определенную величину - шаг;
- число повторений цикла должно быть конечным, однако, не всегда число повторений известно или может быть вычислено заранее. Выход из цикла осуществляется при выполнении некоторых условий. Когда число повторений известно или может быть определено заранее, выход их цикла осуществляется при достижении параметром некоторой заранее заданной величины. Для такого рода задач используется оператор цикла "**FOR".** В общем виде оператор цикла записывается следующим образом:

FOR I = I0 TO IN STEP DN ' начало цикла

операторы " тела цикла "

NEXT I

Здесь:

I - параметр цикла (переменная),

I0 - начальное значение параметра цикла (переменная или число),

IN - конечное значение параметра цикла (переменная или число),

DN - шаг изменения параметра (переменная или число), если шаг равен единице, то его можно опустить.

Этот оператор многократно выполняет операторы "тела цикла", находящиеся между FOR и NEXT для всех значений параметра I от I0 до IN. Структура самого оператора включает и подготовку цикла, и изменение параметра, и проверку условия выхода из цикла.

Проиллюстрируем использование оператора " FOR " примерами:

ПРИМЕР 8.1(задача типа а) Составить схему алгоритма и программу вычисления всех значений функции F(x) для всех значений аргумента x:

F(x) = SIN(x)+COS(x), при $-\pi \le x \le \pi$, шаг изменения аргумента dx=0.1

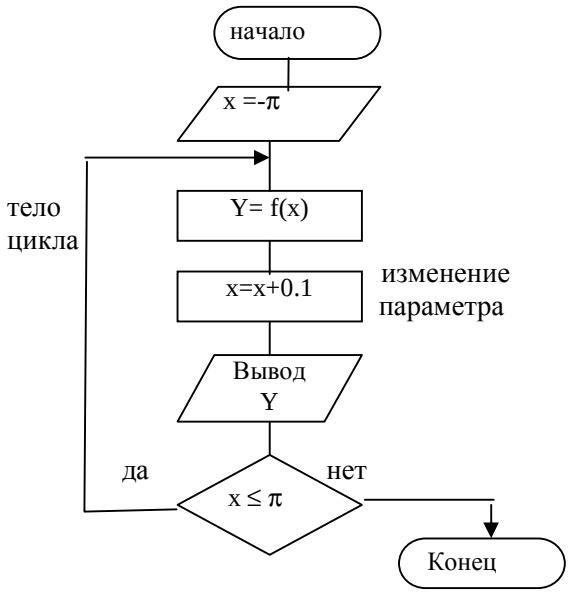


Рисунок 6. Схема алгоритма к задаче 8.1

```
REM  ПРИМЕР 8.1 Циклический алгоритм
   CLS
   CONST PI=3.14
   PRINT  "  X ", " Y "
 FOR  X= - PI TO PI STEP 0.1
      Y= SIN(x)+COS(x)
      PRINT X ,Y
   NEXT X
                END
```

Результат выполнения программы на экране будет иметь вид:

```
X                    Y
-3.14               -1.001591
-3.04               -1.096262
-2.94               -1.179979
-2.84               -1.251906
……….
2.859999            -.6827266
2.959999            -.8029594
3.059999            -.9151694

Press any key to continue
```

Пояснение к программе: в цикле FOR многократно вычисляются значения функции для всех значений аргумента. Значения аргумента и проверка условия выхода из цикла осуществляется самим оператором FOR

ПРИМЕР 8.2 (задача типа б) Вычислить сумму n слагаемых ряда:

$$S = 1 + \frac{1}{3} + \frac{1}{6} + \frac{1}{9} + ... + \frac{1}{3n} = 1 + \sum_{i=1}^{n} \frac{1}{3i}$$

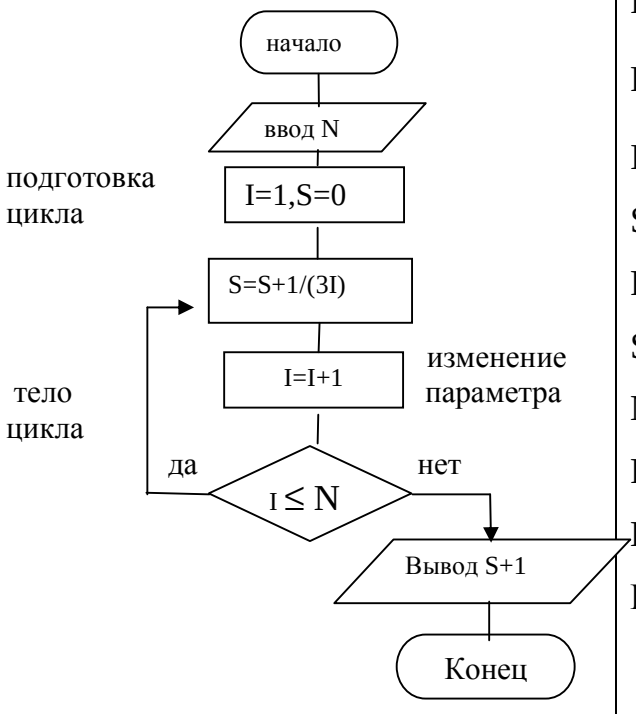


Рисунок 7. Схема алгоритма к задаче 8.2

```
REM Цикл с известным числом
REM повторений
INPUT "Число слагаемых:", N
S = 1
FOR I=1 TO N
S=S+1/(3*I)
NEXT I
PRINT "       Сумма ряда :"; S+1
END
```

Результат выполнения на экране:

Сумма ряда: < число >

ПРИМЕР 8.3 (задача типа б) Вычислить факториал n чисел:
P= n!=1*2*3*….*n

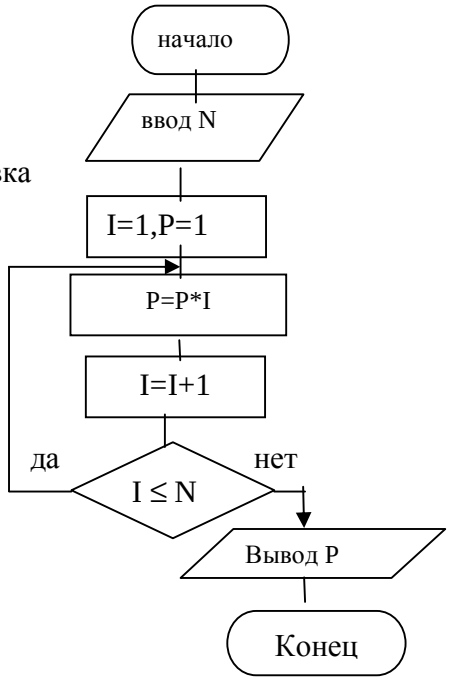


Рисунок 8Схема алгоритма к задаче 8.3

```
REM Цикл с известным числом
REM повторений
INPUT "Число сомножителей:", N
P = 1
FOR I=1 TO N
P=P*I
NEXT I
PRINT "        N факториал :"; P
END
```

Результат выполнения на экране:

N факториал : < число >

<u>Пояснение к программам 8.2, 8.3:</u> С помощью циклов в этих программах последовательно вычисляется значение суммы и факториала (произведения). При этом каждое следующее значение вычисляется через предыдущее. Результатом многократных вычислений является одна величина - сумма ряда заданного числа слагаемых (8.2) или произведение заданного числа сомножителей (8.3).

В приведенных примерах 8.1 - 8.3 число повторений легко определяется заранее. В примере 8.1 параметром цикла является переменная "x", а в 8.2 и 8.3 - переменная "I".

Когда условие задачи не позволяет определить заранее число повторений цикла, выход из цикла зачастую осуществляется по достижении заданной точности вычислений или по какому-либо другому условию, оговоренному в задании. В этих случаях удобно использовать следующие операторы цикла:

а) DO WHILE L < операторы " тела цикла " > **LOOP**	б) **DO UNTIL L** < операторы " тела цикла " > **LOOP**
в) **DO** < операторы " тела цикла " > **LOOP WHILE L**	г) **D0** < операторы " тела цикла " > **LOOP UNTIL L**

Здесь: L - логическое выражение.

Операторы, находящиеся между DO и LOOP повторяются до тех пор, пока выражение, стоящее после WHILE - истинно или до тех пор, пока выражение, стоящее после UNTIL - ложно.

ПРИМЕР 8.4 Вычислить сумму S ряда с заданной точностью e и количество слагаемых N: $S = 1 + \frac{1}{3} + \frac{1}{6} + \frac{1}{9} + ... + = 1 + \sum_{i=1}^{\infty} \frac{1}{3i}$

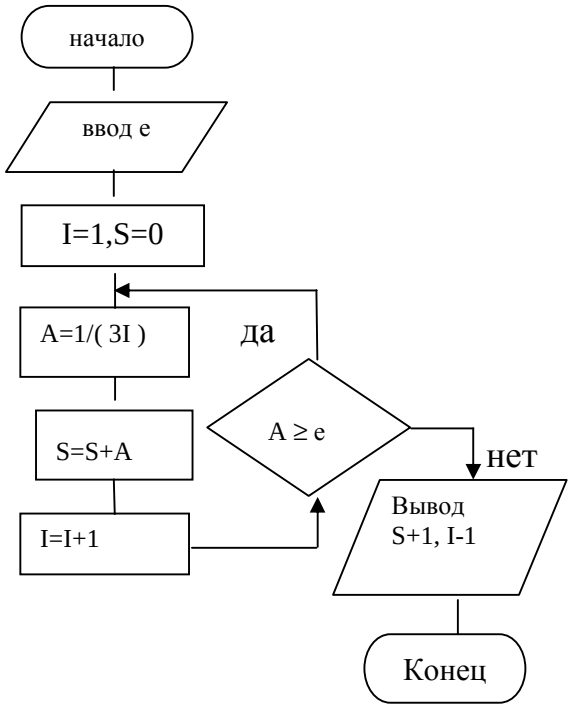


Рисунок 9 Схема алгоритма к задаче 8.4

```
REM Цикл с неизвестным числом
REM повторений
CLS
INPUT "Введите точность", e
S = 0: I=1
DO
A = 1/( 3*I ) ' слагаемое
 S = S+A :    I=I+1
LOOP WHILE A >= e
PRINT "Сумма ряда:"; S+1,"N="; I-1
END
```

Результат выполнения на экране: **Сумма ряда:<число> N=<число>**

Вычислить сумму ряда с точностью **e** - это значит производить вычисления до тех пор, пока очередной член последовательности не станет меньше или равен заданной точности.

В примерах 8.2 и 8.4, в обоих случаях требуется вычислить сумму ряда. В первом случае количество слагаемых определено условием задачи, а во

втором - слагаемое последовательно, в цикле, прибавляется к сумме до тех пор, пока оно (слагаемое) не станет меньше или равно некоторого маленького числа (заданной точности). Такая постановка задачи имеет смысл только для **сходящихся** рядов, т.е. слагаемое должно стремиться к нулю.
Далее приведено еще несколько примеров циклических программ.

ПРИМЕР 8.5 Найти корень уравнения $3x^3 + 5x^2 - 6.9 = 0$ методом деления отрезка пополам с точностью **e** на интервале **[a,b]**

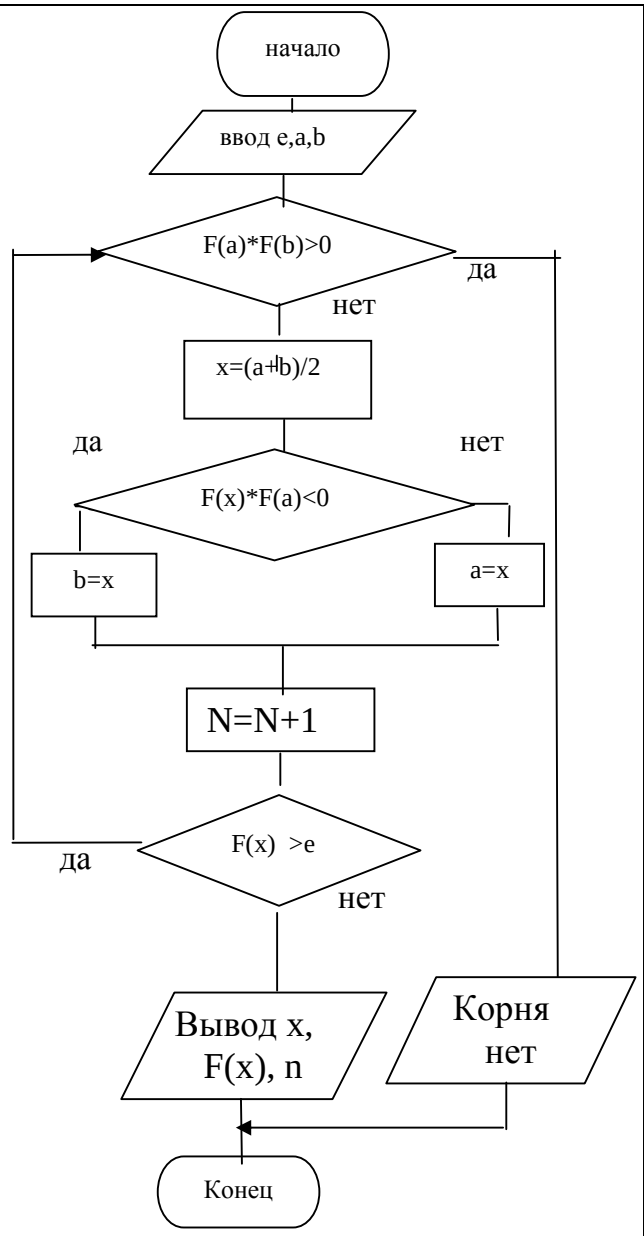


Рисунок 10Схема алгоритма к задаче 8.5

```
CLS
REM Цикл с неизвестным числом
REM повторений
INPUT "Введите точность", e
INPUT "Задайте интервал a,b:",a,b
'Определим значения функции
'на концах интервала
FA = 3 * a ^ 3 + 5 * a ^ 2 - 6.9
FB = 3 * b ^ 3 + 5 * b ^ 2 - 6.9
'если знаки FA и FB одинаковы
' то корня на интервале нет
IF FA * FB > 0 THEN GOTO 100
        DO
x=(a+b)/2  'Делим отрезок пополам
'подставляем значение x
        FX = 3 * x ^ 3 + 5 * x ^ 2 - 6.9
' если корень на отрезке a - x, то
'смещаем интервал влево, иначе -
‘вправо
IF FA * FX < 0 THEN
                        b = x
                ELSE
                        a = x
        END IF
   'считаем число приближений N
        N = N + 1
      LOOP WHILE ABS(FX) > e
    PRINT " корень уравнения:";x
PRINT"FX="; FX, "N=";N
                GOTO 200
100 PRINT "корня не существует"
200 END
```

Пример 8.6 Найти максимум функции Y=SIN(X)*EXP(X+2.5)/(X-0.9) на интервале -5.1 < X < 3

```
CLS
'МАКСИМУМ ФУНКЦИИ ОБОЗНАЧИМ ПЕРЕМЕННОЙ  YMAX
' ЗАДАДИМ НАЧАЛЬНОЕ ЗНАЧЕНИЕ ПЕРЕМЕННОЙ YMAX
YMAX = -1E+19
FOR X = -5.1 TO 3 STEP .5                'СРАВНИМ В ЦИКЛЕ ТЕКУЩЕЕ
Y = SIN(X) * EXP(X + 2.5) / (X - .9)    'ЗНАЧЕНИЕ ФУНКЦИИ Y  С  YMAX
 IF Y > YMAX THEN YMAX = Y        'И ЕСЛИ YMAX ОКАЖЕТСЯ МЕНЬШЕ
                                  'ТО ЕГО  ЗНАЧЕНИЕ  ЗАМЕНЯЕТСЯ НА ТЕКУЩЕЕ
NEXT X
 PRINT
 PRINT "YMAX="; YMAX       ' ВЫВОДИМ ЗНАЧЕНИЕ МАКСИМУМА ФУНКЦИИ
 END
```

Пример 8.7 Определить является ли случайное число X простым?

```
CLS
          ' ЗАДАЕМ  ЦЕЛОЕ ЧИСЛО С ПОМОЩЬЮ ДАТЧИКА СЛУЧАЙНЫХ ЧИСЕЛ
RANDOMIZE TIMER
X=FIX(RND*100)
FOR j = 2 TO X\2
      'В ЦИКЛЕ ОПРЕДЕЛЯЕМ ДЕЛИТСЯ ЛИ ВВЕДЕННОЕ ЧИСЛО НА КАКОЕ-ЛИБО
'ДРУГОЕ  КРОМЕ 1 И САМОГО СЕБЯ БЕЗ ОСТАТКА,
 IF X  MOD  j = 0 THEN  20                    'ЕСЛИ ДЕЛИТСЯ, ТО ОНО НЕ ПРОСТОЕ
NEXT j
PRINT USING " простое число: ####"; X
GOTO 50
20 PRINT X, " - это число не простое"
50 END
```

СЛОЖНЫЕ ЦИКЛЫ

Цикл называется сложным, если он содержит в себе другой, вложенный в него цикл. Количество вложенных друг в друга циклов (глубина вложений) может быть достаточно большим. Каждому циклу соответствует свой параметр. Типы циклов, из которых образован сложный, могут быть различными, это зависит от конкретной задачи. Первоначальный вход в любой цикл допустим только через блок подготовки соответствующего цикла. В общем виде схема алгоритма сложного цикла приведена на рисунке 10. Тело цикла включает в себя операторы соответствующего цикла. Причем для каждого значения параметра внешнего цикла параметр внутреннего цикла пробегает все свои значения.

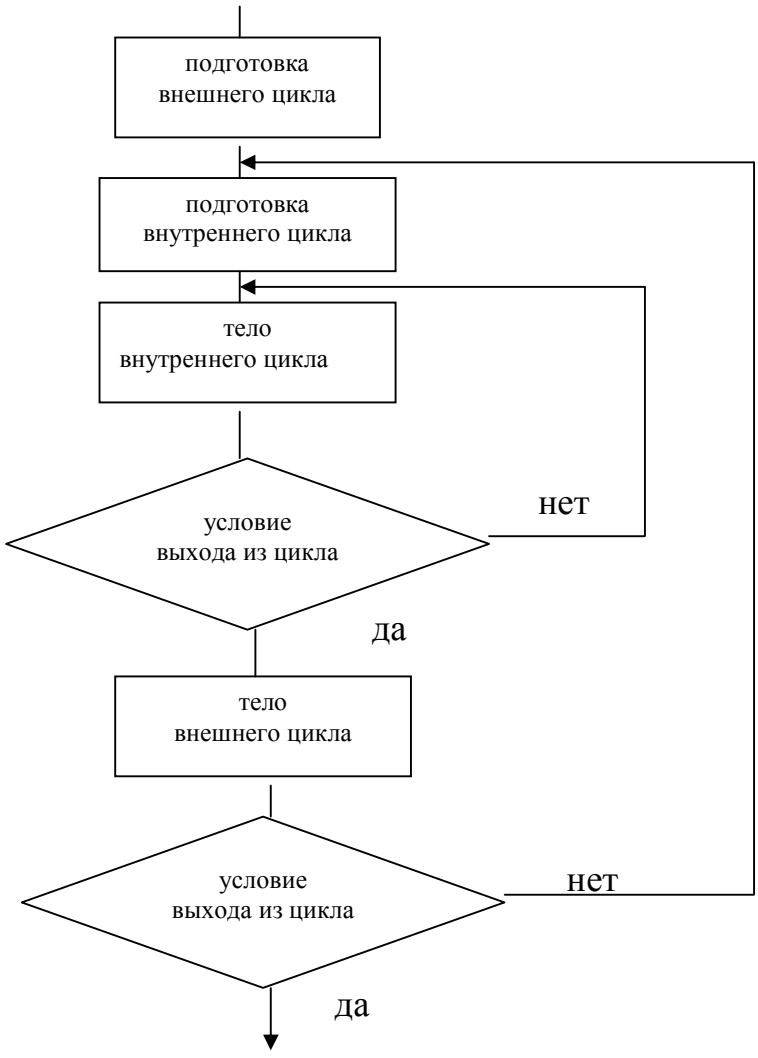


Рисунок 11 Схема алгоритма сложного цикла глубиной два

Если несколько усложнить условия предыдущей задачи, то для ее решения придется использовать алгоритм сложного цикла.

Пример 8.8: Найти и вывести все простые числа от 1 до 1000 (сложный цикл)

```
                'ВЫВЕДЕМ ВСЕ ПРОСТЫЕ ЧИСЛА ОТ 1 ДО 1000
CLS
n = 0
FOR i = 1 TO 1000                       'НАЧАЛО ВНЕШНЕГО ЦИКЛА
FOR j = 2 TO i - 1                      'НАЧАЛО ВНУТРЕННЕГО ЦИКЛА
IF i MOD j = 0 THEN
GOTO 20
END IF
NEXT j                                  'ЗАВЕРШЕНИЕ ВНУТРЕННЕГО ЦИКЛА
n = n + 1
PRINT USING "####"; i;   'ОРГАНИЗУЕМ ВЫВОД ПРОСТЫХ ЧИСЕЛ
IF n MOD 16 = 0 THEN              'НА ЭКРАН ПО 16 ЧИСЕЛ В СТРОКЕ
PRINT
ELSE
END IF
20 NEXT i                               'ЗАВЕРШЕНИЕ ВНЕШНЕГО ЦИКЛА
PRINT " КОЛИЧЕСТВО ПРОСТЫХ ЧИСЕЛ НА ИНТЕРВАЛЕ:"; n
END
```

Пример 8.9 Вывести таблицу значений функции Z, значение которой зависит от двух переменных x, y, которые изменяются пошагово независимо друг от друга, каждая на своем интервале:

$$Z = \frac{\sin xy}{\sqrt[3]{x^2 - y^2}}, \text{ если } a \geq x \geq b,\ \Delta x = m;\ c \geq y \leq d,\ \Delta y = h$$

Для того чтобы найти значения функции при всевозможных сочетаниях значений аргументов, нужно для каждого значения одной переменной (например, x) перебирать все значения второй переменной (например, y). Т.е. необходимо организовать сложный цикл с двумя независимыми параметрами.

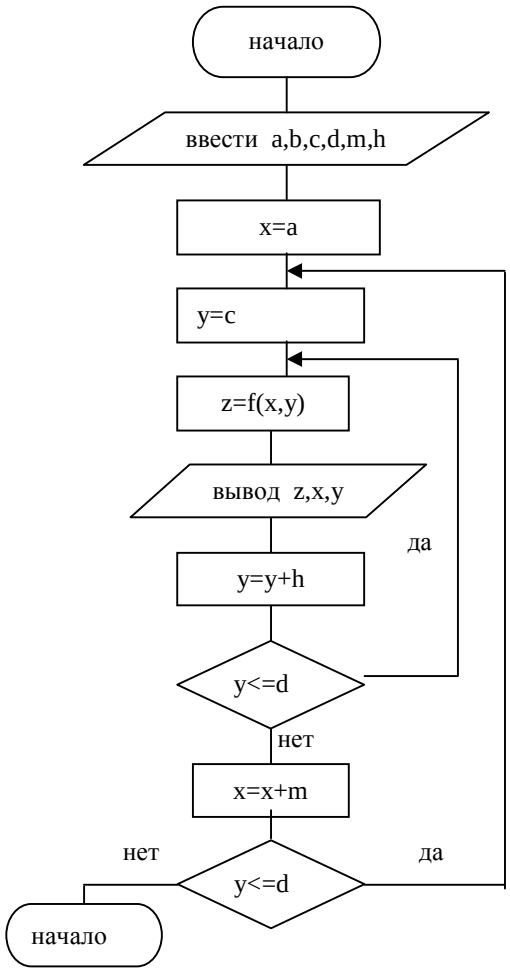


Рисунок 12 Схема к задаче 8.9

```
Rem сложный цикл
INPUT a, b, c, d, m, h
'Выводим заголовок таблицы:
PRINT " Z          X          Y "
    FOR x = a TO b  STEP  m
        FOR y = c TO d  STEP  h
z=SIN(x*y)/(x^2-^2)^(1/3)
            PRINT z, x, y
            NEXT  y
        NEXT  x
                END
```

Индивидуальные задания по лабораторной работе состоит из трех задач.

Далее приведены примеры типовых заданий по программированию циклических алгоритмов.

Задание 1

1. Числа Фибоначчи f_n определяются следующим образом: $f_0=f_1=1$; $f_n=f_{n-1}+f_{n-2}$. Определить сумму чисел Фибоначчи, не превосходящих некоторого заданного числа a.

2. Даны натуральные числа n, m. Определить НОД (наибольший общий делитель) этих чисел с помощью алгоритма Евклида.

3. Дано натуральное число n > 2. Определить все делители этого числа.

Задание 2

Написать программу вычисления и схему алгоритма в двух вариантах:

а) вычислить сумму ряда для заданного количества слагаемых N. Значения переменных задать в диалоговом режиме самостоятельно. На экран вывести значение суммы ряда;

б) вычислить сумму ряда с заданной точностью e. Значения переменных задать в диалоговом режиме самостоятельно. На экран вывести значение суммы ряда и количество повторений

1. $S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \frac{1}{16} - .... + (-1)^n \frac{1}{2^n}$

2. $S = 1 + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \frac{1}{1 \cdot 2 \cdot 3 \cdot 4} + \frac{1}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5} + ... + \frac{1}{N!}$

3. $S = -\frac{2}{4} + \frac{2}{7} - \frac{2}{10} + \frac{2}{13} - \frac{2}{16} + ...$

Задание 3

1. Дано натуральное число n. Вывести такие натуральные неотрицательные числа a, b, c, d, что a^2 + b^2 + c^2 + d^2 = n. (Это справедливо для любого n согласно теореме Лагранжа).

2. Дано натуральное число n. Вывести все СОВЕРШЕННЫЕ числа, меньшие n. (Число является совершенным, если оно равно сумме его делителей кроме себя самого).

3. Даны натуральные числа m и n. Получить все натуральные числа d в диапазоне от m до n, для которых значение функции sin(d) отличается от ближайшего целого числа не более чем на 0.2.

Лабораторное задание

1. Набрать, отладить и выполнить программы, реализующие циклические алгоритмы Вашего индивидуального задания.
2. Создать исполняемые файлы (с расширением exe).
3. Проанализировать работу операторов, пользуясь отладочными режимами.
4. Составить краткий конспект. Защитить работу.

ЛАБОРАТОРНАЯ РАБОТА №9
РАБОТА С МАССИВАМИ В БЭЙСИКЕ

Цель работы:

1. Изучение приемов программирования с использованием массивов.
2. Закрепление навыков работы в отладочных режимах среды QuickBASIC

Массивом называют совокупность данных *одного типа*, обозначаемую одним именем. В зависимости от типа данных массивы могут быть как числовыми, так и текстовыми. При работе с массивами, в ЭВМ под каждый элемент массива отводится ячейка памяти, обращение к которой осуществляется с помощью имени массива с индексом, например A(15). Положение элемента в массиве определяется индексами: одним - для одномерных массивов, двумя - для двумерных (матриц) и т.д. В QuickBASICe допускаются массивы размерностью 255. Максимальное значение каждого индекса не должно превышать 32767.

Имя массива образуется также, как имя простой переменной. Индексы заключаются в круглые скобки и разделяются запятой, если массив не одномерный. В качестве индексов могут быть числа, переменные или арифметические выражения, значения которых автоматически округляются до целого. Если индексы не числовые, то их значения должны быть определены заранее.

Примеры обозначения в Бэйсике элементов массивов:

AQ(33), AQ(I), AQ(I + 4/3) - для одномерного массива;
AD(12,3), AD(I,J), AD(I/2,J+3) - для двумерных массивов.

В том случае, когда какой-либо из индексов массива превышает 10, то такой массив должен быть объявлен оператором DIM заранее.

В операторе DIM указываются имена массивов и в круглых скобках верхние и нижние границы изменения индексов, которые должны быть целыми положительными числами или переменными, значение которых определено в программе ранее.

Если в процессе выполнения программы значение индекса превысит верхнюю границу массива, то система выдаст сообщение ***Subscript out of range*** (Индекс вне диапазона).

Например, **DIM ASD12(5 TO 50)** - оператор описывает одномерный массив, имя которого ASD12, а индексы могут принимать значения от 5 до 50, т.о. под этот массив выделяется 46 ячеек памяти.

Значение нижней границы индексов может быть опущено, и тогда по умолчанию оно принимается равным нулю, например:

DIM MASSIV1(15), MATR2(5,8) - оператор описывает два массива:

- одномерный, имя которого MASSIV1 ,а индексы могут принимать значения от 0 до 15, т.о. зарезервировано 16 ячеек;

- двумерный (матрицу), имя которого MATR2, при этом индекс строки может принимать значения от 0 до 5, а индекс столбца - от 0 до 8, т.о. зарезервировано 64 ячейки памяти.

При обозначении двумерных массивов индекс строки стоит на первом месте, индекс столбца - на втором.

В БЕЙСИКе обработка массивов осуществляется поэлементно, в том числе и ввод - вывод массива. Если массив содержит всего несколько элементов, то задать их значения можно с помощью операторов присваивания:

```
DIM Q(4)
Q(1)=0.25: Q(2)=0.12: Q(3)=0.35: Q(4)=0.28
```

или с помощью оператора ввода:

```
DIM Q(4)
INPUT Q(1), Q(2), Q(3), Q(4)
```

Аналогичным образом осуществляется и вывод массива:

```
PRINT Q(1), Q(2), Q(3), Q(4)
```

В том случае, когда массив содержит много элементов и перечисление их при вводе - выводе становится неудобным, организуется цикл.

Далее приведены фрагменты программ - возможные варианты ввода и вывода значений элементов одномерного массива:

```
REM ВВОД МАССИВА с помощью
' оператора " INPUT "*
INPUT n
DIM Q(n)
FOR I=1 TO n
INPUT  Q(I)
NEXT I
```

```
REM ВВОД МАССИВА с помощью
'оператора " INPUT " из файла
OPEN "d:\dan.dat" FOR INPUT AS #1
INPUT n
DIM Q(n)
FOR I=1 TO n
INPUT #1, Q(I)
NEXT I
```

* По запросу " INPUT" после ввода каждого элемента массива нужно нажать клавишу **Enter**

REM ВВОД МАССИВА с помощью 'оператора " READ "* DATA 5,7,12,2,0,7,23,6,4,8,1 DIM Q(11) FOR I=1 TO 11 READ Q(I) NEXT I	REM ВВОД МАССИВА с помощью ' датчика случайных чисел*** RANDOMIZE TIMER INPUT n DIM Q(n) FOR I=1 TO n Q(I)= FIX(RND*60) NEXT I
'ВЫВОД МАССИВА НА ЭКРАН FOR I=1 TO N PRINT Q(I); NEXT I	'ВЫВОД МАССИВА В ФАЙЛ OPEN "d:\rez.dat" FOR OUTPUT AS #2 FOR I=1 TO N PRINT #2,Q(I); NEXT I

Пример 9.1 Нахождение максимального элемента массива.

```
'НАХОЖДЕНИЕ МАКСИМАЛЬНОГО ЭЛЕМЕНТА МАССИВА
  DIM X(15)
'ВВЕДЕМ ПЕРЕМЕННУЮ YMAX ДЛЯ ХРАНЕНИЯ В НЕЙ ЗНАЧЕНИЯ
'МАКСИМАЛЬНОГО ЭЛЕМЕНТА В МАССИВЕ:
YMAX = -1E+19 'НАЧАЛЬНОЕ ЗНАЧЕНИЕ ПЕРЕМЕННОЙ YMAX
FOR I = 1 TO 15       'В ЦИКЛЕ
   INPUT X(I)       ' ВВОДИМ ЭЛЕМЕНТЫ МАССИВА И СРАВНИВАЕМ ИХ С YMAX
   IF X(I) > YMAX THEN      'И ЕСЛИ YMAX ОКАЖЕТСЯ МЕНЬШЕ
                YMAX = X(I)       ' ТО ЕГО ЗНАЧЕНИЕ ЗАМЕНЯЕМ
                N=I                    ' ЗАПОМНИМ НОМЕР МАКСИМАЛЬНОГО ЭЛЕМЕНТА
     END IF
           NEXT I
             PRINT       ' ПРОПУСКАЕМ СТРОКУ
'ВЫВОДИМ ЗНАЧЕНИЕ МАКСИМАЛЬНОГО 'ЭЛЕМЕНТА МАССИВА И ЕГО НОМЕР:
PRINT "YMAX="; YMAX, "НОМЕР=";N
'ЕЩЕ РАЗ ВЫВОДИМ МАКСИМАЛЬНЫЙ ЭЛЕМЕНТ МАССИВА, НО ПО-ДРУГОМУ:
     PRINT "X(";N;")=";YMAX
       END
```

---

* * Оператору "READ" в программе должен сопутствовать оператор "DATA", в котором перечисляются значения всех элементов массива через запятую

* ** При использовании функции "RND" рекомендуется предварительно запустить датчик случайных чисел "RANDOMIZE TIMER"

Для ввода-вывода двумерных массивов - матриц, организуется сложный (вложенный - глубиной два) цикл:

```
'ВВОД МАССИВА A(n,m)  с помощью
'оператора INPUT, построчно
INPUT n, m
DIM A(n, m)
FOR I=1 TO n
      FOR J=1 TO m
INPUT; A(I,J)
      NEXT J
PRINT
NEXT I
```

```
'ВВОД МАССИВА A(n,m)  с помощью
'оператора INPUT по столбцам
INPUT n, m
DIM A(n, m)
FOR I=1 TO m
      FOR J=1 TO n
INPUT; A(J, I)
      NEXT J
PRINT
NEXT I
```

```
'ВВОД МАССИВА A(n,m)  с помощью
'датчика случайных чисел и вывод
'на экран
RANDOMIZE TIMER
INPUT n, m
DIM A(n, m)
FOR I=1 TO n
      FOR J=1 TO m
   A(I,J) = FIX(RND*50)-20
PRINT A(I, J) ;
      NEXT J
PRINT
NEXT I
```

```
REM ВВОД  МАССИВА с помощью
'оператора " INPUT " из файла и
'вывод на экран
OPEN "d:\dan.dat" FOR INPUT AS #1
INPUT n, m
DIM Q(n, m)
FOR I=1 TO n
FOR J=1 TO m
INPUT #1, Q(I, J)
PRINT Q(I, J) ;
NEXT J
PRINT
NEXT I
```

Пример 9.2: Сформировать одномерный массив из максимальных элементов столбцов матрицы A(22,5). В свою очередь матрицу A получить с помощью датчика случайных чисел.

```
  ' ФОРМИРОВАНИЕ МАССИВА
CLS                    'ОЧИЩАЕМ  ЭКРАН
 DIM A(22, 5), B(5)        'ОПИСЫВАЕМ ИСХОДНЫЙ МАССИВ А И ИСКОМЫЙ-В
     "ЗАПУСКАЕМ ДАТЧИК СЛУЧАЙНЫХ ЧИСЕЛ, ЧТОБЫ СФОРМИРОВАТЬ МАССИВ А
RANDOMIZE TIMER
 FOR I = 1 TO 22
            FOR J = 1 TO 5
 A(I, J) = RND*100                                    'ФОРМИРУЕМ  МАССИВ А
 PRINT USING "####.#####"; A(I, J);      'И ВЫВОДИМ ЕГО ПО СТРОКАМ
            NEXT J
 PRINT
 NEXT I
PRINT
                  'ПЕРЕБИРАЕМ ПОЛУЧЕННУЮ МАТРИЦУ А ПО СТОЛБЦАМ:
 FOR J = 1 TO 5
                   'ИСПОЛЬЗУЕМ ПЕРЕМЕННУЮ AMAX ДЛЯ ХРАНЕНИЯ В НЕЙ
                'ЗНАЧЕНИЯ МАКСИМАЛЬНОГО ЭЛЕМЕНТА СТОЛБЦА МАТРИЦЫ
AMAX = A(1, J)                    'НАЧАЛЬНОЕ ЗНАЧЕНИЕ ПЕРЕМЕННОЙ
                               'AMAX:  - ЗНАЧЕНИЕ ПЕРВОГО ЭЛЕМЕНТА СТОЛБЦА
    FOR I = 2 TO 22
                           'СРАВНИМ ЭЛЕМЕНТЫ СТОЛБЦА С AMAX:
IF AMAX <= A(I, J) THEN AMAX=A(I, J)         'И ЕСЛИ AMAX ОКАЖЕТСЯ
                                      'МЕНЬШЕ, ТО ЕГО  ЗНАЧЕНИЕ  ЗАМЕНЯЕМ
     NEXT I
 B(J) = AMAX                ' ЭЛЕМЕНТУ МАССИВА  В  ПРИСВОИМ  ЗНАЧЕНИЕ
       'МАКСИМАЛЬНОГО ЭЛЕМЕНТА СТОЛБЦА МАССИВА  А
 NEXT J
 FOR J = 1 TO 5                                  'ЦИКЛ ВЫВОДА МАССИВА В
 PRINT USING "####.#####"; B(J);      'ВЫВОД ФОРМАТИРОВАННЫЙ
 NEXT J
 END
```

Пример 9.3: Дан список N учеников класса с указанием фамилии, имени, отчества и даты рождения: число, месяц, год. Вывести (в полном формате) список учеников, родившихся между двумя заданными датами, а также количество учеников, родившихся по месяцам.

Зададим исходные данные с помощью операторов "READ" и "DATA", обозначим исходные массивы:

- fio$ - текстовый массив ФИО;
- d, m, y - массивы дней, месяцев, годов рождения соответственно.

Затем в цикле проанализируем даты рождения всех учеников на вхождение в заданный интервал и, если даты попадают в интервал - записываем в новые массивы: fio1$, d1, m1, y1.

- d0, dn, m0, mn, y0, yn - две заданные даты (начальные, конечные): - числа, месяцы, годы

```
      DATA Асманов Петр Петрович,10,4,1989, Алешина Ирина Сергеевна,23,7,1988
      DATA Алябьева Светлана Ивановна,28,2,1990, Андреев Сергей Семенович,27,4,1989
      DATA  Беликов Илья Михайлович,15,7,1988, Бибаев Виктор Алексеевич,12,7,1990
      INPUT "n<=6: ", n    'вводим количество учеников не больше 6
      INPUT d0, m0, y0, dn, mn, yn     'задаем интервал по датам рождения
      DIM fio$(n), d(n), m(n), y(n)      'описываем массивы
      PRINT:         PRINT "Полный список "; n; " учеников"     ' заголовок
                          ' цикл ввода и вывода списка
    FOR i = 1 TO n
      READ fio$(i), d(i), m(i), y(i) :   PRINT fio$(i), d(i), m(i), y(i)
    NEXT
'в следующем цикле анализируем даты рождения  учеников на вхождение в
 'заданный интервал и формируем новые массивы: fio1$, d1, m1, y1
      DIM fio1$(n), d1(n), m1(n), y1(n)      'описываем новые массивы
  FOR i = 1 TO n
       IF y(i) < y0 OR y(i) > yn THEN GOTO 10
       IF y(i) = y0 AND m(i) < m0 THEN GOTO 10
       IF y(i) = y0 AND m(i) = m0 AND d(i) < d0 THEN GOTO 10
       IF y(i) = yn AND m(i) > mn THEN GOTO 10
       IF y(i) = yn AND m(i) = mn AND d(i) > dn THEN GOTO 10
       k = k + 1 'считаем количество, попадающих в интервал
       fio1$(k) = fio$(i):   d1(k) = d(i):   m1(k) = m(i):     y1(k) = y(i)
10 NEXT
IF k <> 0 THEN      'если имеются ученики попадающие в интервал, то печатаем их ФИО
       PRINT:     PRINT "ученики, родившиеся между заданными датами:"
  FOR i = 1 TO k
        PRINT fio1$(i), d1(i), m1(i), y1(i)
  NEXT
END IF
  IF k = 0 THEN
  PRINT:   PRINT "учеников, родившихся между":  PRINT "двумя заданными датами - нет"
  END IF
       PRINT:      PRINT "Количество учеников,  родившихся по месяцам:"
'циклы, определяющие сколько учеников родилось в одном из 12 месяцев
    FOR L = 1 TO 12
      v = 0
       FOR i = 1 TO n
            IF m(i) = L THEN v = v + 1
       NEXT
      IF v <> 0 THEN PRINT v; "уч. род. в "; L; "месяце"
    NEXT
```

Индивидуальное задание по лабораторной работе состоит из трех задач. Далее приведены примеры типовых заданий по программированию циклических алгоритмов с использованием массивов.

Задание 1

1. Дан массив целых чисел X(50). Сформировать из него массив Y(50), в котором первыми располагаются четные элементы массива X, а затем - нечетные элементы. Вывести оба массива.
2. Дан массив целых чисел X(50) и число K. Сформировать массив Y(50-i) элементов после i-го, (i - номер максимального элемента X среди x1,...xK). Вывести оба массива.
3. Дан массив целых чисел X(50). Определить в нем число пар, элементы которых имеют разные знаки и первый больше второго. Вывести исходный массив, пары элементов и результат.

Задание 2

1. Дана матрица X(n, m) целых чисел. Обнулить элементы ее главной диагонали, расположенные в столбцах с четной суммой элементов и строках с нечетной суммой элементов. Вывести исходную и полученную матрицы.
2. Дана матрица X(n, m) целых чисел. Обнулить те ее столбцы, в которых элементы, принадлежащие одной из главных диагоналей, являются максимальными (сами максимальные элементы не обнулять). Вывести исходную и полученную матрицы.
3. Дана матрица X(n, m) целых чисел. Получить новую матрицу, элементы которой являются разностями суммы элементов соответствующей строки исходной матрицы и соответствующего элемента исходной матрицы. Вывести исходную и полученную матрицы.

Задание 3

1. Дан список N учеников класса с указанием фамилии, имени, отчества и даты рождения: число, месяц, год. Вывести (в полном формате) список учеников, родившихся между двумя заданными датами, а также количество учеников, родившихся по месяцам.
2. Дан список N торговых отделов, торгующих N однотипными товарами с указанием дневной выручки по каждому товару. Вывести номера отделов, получивших выручку больше заданной, а также номера отделов, получивших наибольшую выручку (по каждому товару в отдельности).
3. Дан список N сотрудников с указанием фамилии, точной даты рождения, стажа работы и заработной платы. Вывести: список сотрудников, получающих зарплату выше средней по коллективу, упорядоченный по убыванию заработной платы (в полном формате, с указанием всех данных по каждому сотруднику).

ЛАБОРАТОРНОЕ ЗАДАНИЕ

1. Набрать и выполнить на ЭВМ примеры, приведенные в описании. Проанализировать работу операторов, пользуясь отладочными режимами. С помощью отладочных окон режима "Debug" просмотреть промежуточные значения переменных.
2. Создать исполняемые файлы (с расширением exe).
3. Выполнить на ЭВМ программу индивидуального задания результаты вывести на экран и в файл.
4. Написать отчет (краткий конспект и текст программ).

ЛАБОРАТОРНАЯ РАБОТА № 10
СТРОКОВЫЙ ТИП ДАННЫХ В БЭЙСИКЕ.

Цель работы:

1.Изучение приемов программирования с использованием строковых данных.
2.Приобретение практических навыков в работе со строковыми данными.

Описание и ввод строковых данных.

Ранее упоминалось, что в языке БЭЙСИК существует строковый тип данных для обработки последовательности символов. Данными строкового типа являются строковые константы и строковые переменные.

Строковая **константа** представляет собой произвольную последовательность символов, заключенную в двойные кавычки, длиной до 32567 символов, например, " Hello", "Добрый день".

Строковые **переменные** бывают переменной или фиксированной длины. Строка переменной длины (STRING) представляет собой последовательность длиной до 32567 символов из таблицы ASCII. В памяти под такую символьную переменную отводится количество байт равное количеству символов переменной плюс 4. Объявить строковый тип переменной длины можно одним из приведенных ниже способом:

1) явно - с помощью суффикса $: Hello$ = "Привет"
2) явно - с помощью операторов описания типа:

```
DIM Hello AS STRING
```

3) неявно - с помощью оператора объявления типа данных:

```
DEFSTR H ' с буквы H начинаются имена
' переменных типа STRING
Hello = "Привет"
```

Строка фиксированной длины (STRING * N) представляет собой строку длиной N символов. В памяти под такую символьную переменную отводится N байт. Описать символьную переменную фиксированной длины можно таким образом: DIM Hello AS STRING*12

```
Hello = "Привет - Hello"
PRINT " Результат:" ; Hello
```

На экран будет выведено: Результат: Привет - Hel

Строковой переменной можно присвоить значение либо с помощью оператора присваивания, например,

St$ ="abcde",

либо с помощью операторов ввода, например:

```
  DATA "abcde"
READ St$
```

или

```
INPUT St$
```

Однако, при вводе значения строковой переменной оператором INPUT возникают определенные трудности. Если среди символов вводимой строки встречаются запятые, то запятая воспринимается оператором INPUT, как разделитель. Чтобы обойти эту проблему, предусмотрена модификация оператора INPUT:

```
LINE INPUT St$
```

Этот оператор предназначен специально для ввода в одну строковую переменную полной строки текста независимо от ее содержания.

Работа со строками. Строковые операции.

Строковые выражения используются в различных операторах языка Бэйсик: присваивания, условного перехода, вывода и т.д.

Строковое выражение может содержать строковые константы, строковые переменные, вызов функций и строковые операции.

1. Операция **"+"** (конкатенация) предназначена для объединения строк. Результат операции имеет строковый тип. Например, после выполнения фрагмента программы:

```
L$ = "MOSCOW"
AGE$ = "We" + " live in " +  L$
```

строковая переменная AGE$ примет значение:

We live in MOSCOW.

2. Операции сравнения (=, <>, <, >, <=, >=). Сравнение двух строк выполняется слева направо с учетом кодов ASCII. Т.е. сравниваются сначала коды первых символов, затем вторых и т.д. Результат операций сравнения имеет логический тип, то есть принимает значения **ДА** или **НЕТ**, например,

"A" < "B" (результат ДА)

"RA" > "RR" (результат НЕТ)

"2" > "12" (результат ДА)

Если две строки имеют различную длину, но их начальные символы совпадают, включая последний символ более короткой строки, то короткая строка считается меньшей, например:

"12.0" > "12" (результат ДА).

Строки считаются равными тогда и только тогда, когда имеют одинаковую длину и одинаковую последовательность символов, например:

"TURBO" = "TURBO" (результат ДА)

"TURBO" = " TRUBO " (результат НЕТ).

Строковые функции и операторы

Приведем наиболее часто употребляемые строковые функции:

– **ASC(St$)** в качестве результата дает числовое значение ASCII-кода первого символа в символьном выражении.

– **CHR$**(код) в качестве результата дает символ ASCII, код которого является аргументом.

– **LEN(St$)** в качестве результата дает целое число, равное длине строкового выражения St$.

– **MID$(St$, n, m)** в качестве результата дает фрагмент строки St$, длиной m, начиная с позиции n.

– *Оператор* **MID$(St$, n, m)= Zt$** заменяет фрагмент строки St$ длиной m, начиная с позиции n на строку Zt$.

– **STR$**(числовое выражение) - преобразует числовое выражения в символьное. Если его значение положительно, то к полученной строке слева добавляется пробел.

– **VAL(St$)** - преобразует строку в числовое выражение. Функция VAL является обратной по назначению функции STR$. Если первый символ аргумента - не числовой, то функция VAL возвращает ноль. Если первый символ - знак минус, то ликвидируются все левые пробелы. Если первый символ цифра, то остается один левый пробел.

– **LEFT$(St$, n)** - выделяет из выражения n левых символов.

– **RIGHT$(St$, n)** - выделяет из выражения n правых символов.

– **LTRIM$(St$)** - удаляет левые пробелы.

– **RTRIM$(St$)** - удаляет правые пробелы.

– **INSTR(n, St$, Zt$)** - определяет позицию вхождения выражения Zt$ в St$, начиная с n.

– **INKEY$** - функция анализирует информацию о нажатых клавишах. Эта функция, помещенная в цикл DO, может быть использована для создания паузы произвольной продолжительности, управления графическим объектом (пример 11.9), выхода из цикла и т.д.

Программа иллюстрирует работу строковых функций и операций.

```
CLS
READ a$           ' ввод символьной переменной
DATA "символ"      ' значение переменной - " символ"
LINE INPUT b$      ' ввод символьной строки "СИМВОЛ, СТРОКА
q$ = a$ + " ,строка"    ' операция  конкатенации
                PRINT a$, b$, q$
w = ASC(a$)        ' функция определения кода символа
d$ = CHR$(68)      ' функция определения символа по коду
                PRINT w, d$
IF "ff " < "ff" THEN    ' операция сравнения
                    PRINT "ff_ <  ff "
ELSE PRINT "ff_ >= ff"
END IF
IF "ff" > " ff" THEN
                     PRINT "ff > _ff"
ELSE PRINT "ff <= _ff"
END IF
dl = LEN(a$ + b$)        ' функция определения длины
              PRINT dl
c$ = MID$(b$, 5, 3)       ' функция вырезки из выражения
                                        ' 3-х символов, начиная с 5-го
              PRINT c$
MID$(b$, 4, 4) = "птом"   ' замена 4-х символов, начиная  с 4-го на "птом"
              PRINT b$
MID$(b$, 4, 4) = "вол,"     ' обратная замена
              PRINT b$
n$ = STR$(125)      ' преобразование числового  выражения в символьное,
                    ' при положительном числе  слева добавляется пробел
              PRINT n$
v = VAL("  -666")   'преобразование символьного выражения
                     'в числовое   (функция обратная STR$  )
         PRINT v
k$ = LEFT$(n$, 3)  ' выделение из выражения 3-х левых символов
  PRINT k$
l$ = LTRIM$(n$)              ' удаление левых пробелов
         PRINT l$
a$ = "СТРОКА"
m = INSTR(1, b$, a$)       ' определение позиции вхождения строки a$ в b$
        PRINT m
```

Результат выполнения программы:

```
символ СИМВОЛ, СТРОКА символ ,строка
 225 D
ff_ >= ff
ff > _ff
 20
ОЛ,
СИМптом СТРОКА
СИМвол, СТРОКА
 125
-666
 12
125
 9
```

Символ в тексте определяется номером позиции, которую он занимает; переход к следующему символу осуществляется изменением номера позиции на 1. Выбирать из текста можно как по одному символу, так и по несколько следующих друг за другом. Тексты могут содержать и числовую информацию, заданную в символьной форме.

Пример 10.1 Дан текст. Удалить из текста повторяющиеся символы.

```
CLS
t$ = "ббааабочччкаа"    'Исходный текст
PRINT t$
dl = LEN(t$)             'Определяем длину строки
 i = 1                   ' Считаем символы
 DO
s$ = MID$(t$, i, 1)          ' выделяем каждый символ
ss$ = MID$(t$, i + 1, 1)     ' и следующий
IF s$ = ss$ THEN             ' если они совпадают, то
'левую часть строки, включая i-й символ, объединяем с правой, исключая (i+1)-й
t$ = LEFT$(t$, i) + RIGHT$(t$, dl - i - 1)
dl = dl - 1   ' длина уменьшится
i = i - 1      ' будем сравнивать в следующий раз тот же i-й символ с новым (i+1)-м
END IF
i = i + 1
LOOP WHILE i < dl ' перебираем все символы по всей длине строки
PRINT t$  :   END
```

На экране будет: **бабочка**

Пример 10.2 Дан текст. Найти наибольшее количество цифр, идущих подряд.

```
 'Пусть  i - позиция символа в тексте
 '      L - максимальное количество цифр в тексте, идущих подряд
 '     m - счетчик идущих подряд цифр в группе
CLS       '
 PRINT "Введите текст"
 INPUT t$
PRINT "Исходный текст:": PRINT t$
 m = 0      'начальное значение количества  цифр в каждой группе
  L = 0     'начальное значение  кол-ва цифр в наибольшей группе
j = LEN(t$)   'длина строки
  FOR i = 1 TO j
    a$ = MID$(t$, i, 1)       '  выделение каждого символа
FOR k = 0 TO 9        ' цикл сравнения каждого символа с цифрой:
IF a$ = LTRIM$(STR$(k)) THEN  'если текущий символ является цифрой,
 m= m + 1: GOTO 200           'то посчитаем его и выходим из цикла
                              'и берем следующий символ уже во внешнем цикле
END IF
NEXT k
m = 0           ' если же этот символ не цифра то начнем считать сначала
200 IF m > L THEN L = m   'запоминаем число цифр, если оно больше L
 NEXT i
      PRINT "Наибольшее количество цифр в тексте = "; L
END
```

Результат:

```
Исходный текст:
asd123d343434dd34vv6876543321321rtr555rtyhgf6667u7uyuy888888123
Наибольшее количество цифр в тексте = 13
```

Пояснение к программе: Данная задача - есть задача нахождения максимума, поэтому под него выделяем ячейку памяти "L". После ввода текста просматриваем посимвольно текст и каждый символ проверяем на совпадение с цифрой во вложенном цикле с параметром k. При выявлении цифры к текущему значению счетчика m прибавляется 1. Затем сравниваем текущее значение счетчика с содержимым ячейки "L" и если m больше L, то заменяем значение ячейки "L" на большее.

В том случае, когда очередной символ не цифра, мы обнуляем "m".

Просматриваем посимвольно текст до его окончания.

Пример 10.3: Сложить два целых положительных числа большой длины.

```
CLS
INPUT "введите первое число", a$:        INPUT "введите второе число", b$
la = LEN(a$): lb = LEN(b$)
IF la < lb THEN
DO UNTIL la = lb  " цикл выравнивания строк  по длине
a$="0"+a$:  la = LEN( a$)
LOOP
END IF
IF la > lb THEN
DO UNTIL la = lb   ' цикл выравнивания строк  по длине
b$="0" + b$:  lb = LEN(b$)
LOOP
END IF
per = 0
FOR i = la TO 1 STEP -1
s1 = VAL(MID$(a$, i, 1))  ' выделяем по одному символу и преобразуем в число
s2 = VAL(MID$(b$, i, 1))
s = s1 + s2 + per                 'складываем поразрядно числа и перенос
IF s > 9 THEN
ed = s MOD 10: per = 1        'разделяем сумму  на единицы и десятки-перенос
ELSE
ed = s: per = 0
END IF
c$ = LTRIM$(STR$(ed)) + c$          'преобразуем число в строку, убираем пробелы
                                                  'и присоединяем  к результату
NEXT
IF per = 1 THEN c$ = STR$(per) + c$                ' присоединяем последний перенос
PRINT a$:   PRINT b$:   PRINT c$
 END
```

Пояснения к программе*:* Т.к. диапазон представления чисел ограничен, для операций над ними используются символьные данные. С их помощью имитируют арифметические действия над числами большой величины. В данной программе имитируется сложение двух чисел столбиком. Вводятся два числа в символьной форме. Сначала строки выравниваются по длине, для этого к началу более короткой строки приписываются нули. Затем, начиная с конца строк, выделяется по одной цифре из каждой строки и складываются (s1 и s2). Если их сумма больше 9, то отдельно записываются единицы в ячейку "ed" и перенос - в ячейку "per". Перенос участвует в сложении цифр следующего разряда заданных чисел.

Пример 10.4: В тексте определить количество предложений, наибольшее количество слов в предложении и номер этого предложения. (При вводе текста в конце предложений ставить точку и один пробел.)

```
CLS
DATA В Московском государстве жил был Царь. Был у Царя Стрелец молодец. И был еще при Царе хитрый Советник. Был Министр финансов.
READ t$
PRINT  "Исходный текст"
PRINT  t$
k = 0    'считаем кол-во предложений
 j = LEN(t$)   'длина строки
 i1 = 1     'начало предложения
 L = 0     'максимальное кол-во слов - начальное значение
DO
      n = INSTR(i1, t$, ".")    'конец предложения, позиция точки
m = 1        'счетчик слов
k = k + 1
' Во внутреннем цикле считаем количество слов в k- м предложении :
        FOR i = i1 TO n ' от начала предложения до его конца
            a$ = MID$(t$, i, 1)            '  выделяем каждый символ
            b$ = MID$(t$, i + 1, 1)       '  и следующий за ним
IF a$ = " " AND b$ <> " " THEN 'если текущий символ пробел,
                                                'а следующий - нет
             m = m + 1                  '  то начинается новое слово
END IF
            NEXT i
      IF m > L THEN L = m    ' если количество слов в этом
                        ' предложении больше чем L, то запоминаем его
  i1 = n + 2             ' позиция первого слова в следующем  предложении
WHILE    i1 < j          'проверяем не закончился ли текст
PRINT "Наибольшее количество слов в предложении = "; L
PRINT "Количество предложений="; k
END
```

Файл результата:

```
Исходный текст
в Московском государстве жил был Царь. Был у Царя Стрелец молодец.
И был еще при Царе хитрый Советник. Был Министр финансов.
Наибольшее количество слов в предложении = 7
Количество предложений= 4
```

Группу строк (символов) можно хранить в массиве, например A$(i), и обращаться к отдельным строкам по именам A$(1),A$(2),A(3) и т.д. Например, список фамилий в телефонном справочнике можно представить, как одномерный строковый массив.

Для массивов строк допустимы те же имена, что и для числовых массивов, только к имени добавляется знак $ или в описании указывается тип. Каждый элемент массива равноценен строковой переменной. Действие всех вышеперечисленных строковых функций распространяется и на элементы массивов.

Пример 10.5: С клавиатуры вводится список фамилий. Распечатать список в алфавитном порядке.

```
' Задача представляет собой  задачу сортировки одномерного массива
CLS
INPUT n
DIM fio(n) AS STRING
FOR i = 1 TO n
INPUT fio(i)
 NEXT
 FOR i = 1 TO n - 1
 FOR j = 1 TO n - i
' Если  фамилии не по алфавиту, то меняем  их местами
' с помощью оператора "SWAP"
 IF fio(j) > fio(j + 1) THEN SWAP fio(j), fio(j + 1)
 NEXT
 NEXT
 FOR i = 1 TO n
 PRINT fio(i)
 NEXT
 END
```

Пояснение к программе: оператор SWAP A,B - обменивает величины двух переменных одного типа.

Индивидуальное задание по лабораторной работе состоит из трех задач. Далее приведены примеры типовых заданий по программированию задач с использованием строковых данных.

Задание 1

1. Даны строки a$ и b$. Определить, имеются ли в строке a$ все символы, из которых состоит строка b$ и вывести их.
2. В исходной строке a$ определить наличие и количество пар соседних одинаковых символов и удалить их. Вывести строку a$ до и после модификации, а также найденное число пар.

3. В исходной строке a$ (состоящей из цифр) определить сумму цифр, входящих в строку. Вывести строку a$ и результат.

Задание 2

1. В исходной строке a$ заменить каждое нечетное вхождение заданного сочетания символов x$ на соответствующее число пробелов. Вывести исходную и полученную строки.
2. В исходной строке a$ удвоить символы, входящие в первую половину алфавита и удалить все символы второй половины алфавита. Вывести строку до и после изменения.
3. В исходной строке a$ переместить символы следующим образом: первый, последний, второй, предпоследний, и т.д. Вывести строку до и после изменения.

Задание 3

1. Из заданной в виде строки a$ совокупности чисел с дробной частью (разделитель целой и дробной части - точка, разделитель чисел - пробел) определить сумму целых частей чисел и вывести в текстовой форме.
2. Написать программу сложения и умножения восьмеричных чисел с плавающей запятой. Исходные числа и результат должны иметь следующую форму: "0.nnnnn*8^mmm".
3. Рассортировать слова исходной фразы a$ по алфавиту (по заданному номеру N буквы в слове).

Лабораторное задание

1. При домашней подготовке составить программы на языке БЭЙСИК согласно варианту задания.
2. В системе QuickBASIC создать файлы программ.
3. Отладить и выполнить программы. Результат вывести на экран.
4. Проанализировать работу операторов и символьных функций.
5. Написать отчет. (Краткий конспект и распечатки программ и результатов.)

ЛАБОРАТОРНАЯ РАБОТА №11
РАБОТА В ГРАФИЧЕСКОМ РЕЖИМЕ

Цель работы:

1.Изучение приемов программирования с использованием графического режима.

2.Приобретение практических навыков работы в графическом режиме.

Для воспроизведения графики компьютер снабжен специальными аппаратными средствами. К ним относятся монитор и специальное устройство - видеоадаптер (видеокарта), выполняющее роль переводчика

между памятью и экраном. Видеоадаптер вместе с монитором образуют видеоподсистему.

Видеоподсистемы работают в двух видеорежимах: текстовом или графическом. В текстовом режиме экран монитора разбивается на отдельные символьные позиции, в каждой из которых может выводиться только один символ.

В графическом режиме для каждой точки изображения, называемой пикселем, отводится от одного (монохромный режим) до 24-бит (цветной). В этом режиме имеется доступ к каждой точке изображения. Любое изображение можно представить в виде множества мельчайших точек, каждой из которых сопоставлены две координаты и номер цвета. Полученный числовой набор, называемый **растром**, более или менее точно опишет изображение. Графические режимы используются для формирования рисунков.

В программировании используется такая характеристика как *разрешение.* Для графических режимов - это количество доступных точек на экране, для текстовых - количество символов в строке. *Разрешение экрана* является одним из важнейших параметров видеоподсистемы. Чем оно выше, тем больше информации можно отобразить на экране.

Количество различных цветов (ц*ветовое разрешение)*, доступных для раскрашивания изображений - другое важное свойство графического режима. Базовая палитра IBM - совместимых ЭВМ включает 16 стандартных цветов. В программах цвета задаются своими номерами, приведенными ниже:

Номера экранных цветов

номер	**цвет**	**номер**	**цвет**
0	черный	8	серый
1	голубой	9	ярко-голубой
2	зеленый	10	ярко-зеленый
3	бирюзовый	11	ярко-бирюзовый
4	красный	12	ярко-красный
5	розовый	13	ярко-розовый
6	коричневый	14	желтый
7	белый	15	ярко-белый

Современные персональные компьютеры комплектуются дисплеями способными отображать палитры от 256 до 16 млн. цветов.

В графическом режиме каждый пиксель определяется цветом и своими координатами - положением относительно левого верхнего угла экрана, который , в свою очередь, имеет координаты 0,0. Программист может управлять цветом любого пикселя, что позволяет формировать на экране любые изображения, в том числе рисунки, графики, чертежи, символы.

В текстовых режимах можно задавать координаты символа, определяя положение курсора, относительно левого верхнего угла экрана (1,1), цвет символа (цвет переднего плана) и цвет фона (цвет заднего плана).

Функции и операторы графического режима

1. Оператор **CLS** - очищает экран дисплея;
2. Оператор **VIEW [[SCREEN](x1,y1)-(x2,y2)[,[n][,m]]]** - устанавливает окно графического вывода, ограниченное прямоугольной областью с координатами (x1,y1) левого верхнего угла и (x2,y2) правого нижнего угла.

n - числовое выражение: цвет закраски окна;

m - числовое выражение: цвет рамки окна вывода;

SCREEN - аргумент. Определяет, что координаты x, y любой выводимой точки имеют абсолютные значения. Графика выводится только внутри окна. Если SCREEN отсутствует, то внутри окна вывода действуют относительные к границам окна (x1,y1) координаты.

3. Оператор **WINDOW [[SCREEN](x1,y1)-(x2,y2)]-** устанавливает размеры текущего окна вывода. (x1,y1)-(x2,y2) - числа обычной точности, определяющие координаты прямоугольного окна вывода.

Оператор WINDOW позволяет создать координатную систему для рисования линий, графиков и пр. без задания абсолютных координат на экране. Т.о. координаты всех точек определяются как относительные к данному окну вывода.

Модификация WINDOW SCREEN преобразует направление координаты Y в декартово представление, т.е. значения Y идут от большего к меньшему сверху вниз.

Пример 11.1 Сформировать окно на экране (40 <= $x_{экр}$<= 200, 80 <= <= $y_{экр}$ <= 125), соответствующее геометрической области (-5 <= x <= 5, -2 <= y <= 2). (Использование операторов WINDOW и VIEW).

```
SCREEN 9 'Задание разрешения экрана 640 x 350
REM Задание размеров окна на экране в пикселях
   VIEW (40, 80)-(200, 125) , 3 , 7
REM Задание геометрических размеров окна
   WINDOW (-5, -2)-(5, 2)
   END
```

В результате выполнения программы на черном экране, слева отобразится прямоугольник бирюзового цвета в белой рамке.

4.Оператор **SCREEN n**, где **n** - выражение или константа целого типа, указывает номер режима экрана;

Оператор SCREEN используется для выбора режима экрана в соответствии с особенностями аппаратуры видеоадаптера и монитора.

Этот оператор в программе должен предшествовать любым графическим операторам.

Параметры основных режимов оператора **SCREEN** адаптера **VGA**:

Режимы (n)	Разрешение	Число атрибутов	Число цветов	Примечание
0	текстовый режим (обычно 25 строк по 80 символов)			
1	320 x 200	4	16	
2	640 x 200	2	16	
7	320 x 200	16	16	
8	640 x 200	16	16	
9	640 x 350	16	64	
11	640 x 480	2	256	Необходимо применение оператора PALETTE
12	640 x 480	16	256	
13	320 x 200	256	256	

Пример 11.2 Нарисовать прямоугольник и заштриховать его в клетку.

```
CLS
SCREEN 11    ' разрешение 640*480
LINE (0, 0)-(250, 250), , B  ' рисуется прямоугольник
FOR i = 0 TO 250 STEP 10
LINE (i, 0)-(i, 250)      '  вертикальные линии
LINE (0, i)-(250,  i)    '  горизонтальные линии
NEXT
END
```

Горизонтальные линии характерны тем, что координаты по оси Y начальной и конечной точки равны.

Вертикальные линии характерны тем, что координаты по оси X начальной и конечной точки равны.

В том случае, если необходимо чтобы штриховка была под углом 45°, то одинаковыми должны быть разноименные координаты (Пример 11.3)

Пример 11.3 Нарисовать прямоугольник и заштриховать его под углом 45°

```
CLS
SCREEN 11
LINE (0, 0)-(250, 250), , B
FOR i = 0 TO 250 STEP 10
LINE (250, i)-(i, 250) ' линии рисуются вправо от диагонали к нижнему углу
LINE (0, i)-(i, 0) ' линии рисуются вправо от верхнего правого угла
NEXT
```

5.Оператор **COLOR [n],[m]** - устанавливает экранные цвета;

где n - цвет переднего плана (букв или графич. изображений);

m - цвет фона.

Например, COLOR 10,4 - оператор установит на красном фоне ярко-зеленый цвет букв или графических изображений.

6. Оператор **PALETTE [n,m]** - изменяет цвет в палитре,

где n - атрибут палитры, который нужно изменить(0 - 15);

m - цвет, присваиваемый атрибуту. Величина длинного, целого типа.

PALETTE USING A%[(i)] - изменяет все атрибуты одновременно, где A% - имя массива длинного, целого типа. Массив A% должен содержать номера цветов, присваиваемых атрибутам текущего режима экрана.

i - индекс начального элемента массива, используемый для установки палитры.

7. Операторы **PRESET [STEP](x,y)[,n]; PSET [STEP](x,y)[,n]** - рисуют точку на экране,

где (x,y) - координаты точки на экране. Величины координат отображаемых на экране точек зависят от установленного графического режима (оператор SCREEN). Значения координат могут выходить за пределы экрана, но не должны превышать значений (от -32768 до 32767);

STEP - указывает, что координаты (x,y) отображаемой точки берутся как приращение к соответствующим координатам предыдущей точки. При отсутствии аргумента STEP координаты (x,y) непосредственно указывают физические координаты экрана;

n - цвет.

Различие операторов PSET и PRESET состоит в том, что если цвет опущен, в первом случае точка выводится цветом переднего плана, а во

втором - цветом фона. На атрибут цвета влияют операторы COLOR и PALETTE.

Пример11.4 Сформировать на экране график двух периодов функции sin(x). (Использование операторов WINDOW, VIEW,SCREEN, COLOR, PSET):

```
 REM ПОСТРОЕНИЕ СИНУСОИДЫ
SCREEN 9       ' Задание разрешения экрана 640 x 350
 pi = 3.1415       ' Задание числа "пи"
  COLOR 12      ' Задание цвета
  '  выделяем часть экрана для построения графика:
VIEW (20, 50)-(639, 349)
'  установим удобную систему координат:
 WINDOW SCREEN (-2 * pi - 1, 1.1)-(2 * pi + 1, -1.1)
 FOR i = 0 TO 4 * pi STEP pi / 32000
 REM Формирование синусоиды по точкам
 PSET (i - 2 * pi, SIN(i))
 NEXT i
   END
```

В результате выполнения программы на экран будет выведена синусоида красного цвета.

8. Оператор **LINE [[STEP](x1,y1)]-[STEP](x2,y2)[,[n][,[B[F]][,&m]]]** рисует линии, контуры прямоугольников, закрашенные прямоугольники.

(x1,y1)-(x2,y2) - устанавливают координаты начала и конца отрезка;

n - цвет линии, контура или закраски прямоугольника;

B- параметр устанавливает режим рисования прямоугольника;

BF- параметр устанавливает режим рисования прямоугольника, закрашенного цветом n;

STEP - указывает на относительные координаты (относительно последней точки).

&m - параметр "стиль" - 16-битовое целое число. Используется для рисования пунктирных линий.

9. Оператор **CIRCLE [STEP](x,y),r[,[n][,[fi1][,[fi2][,k]]]]** - рисует эллипс или окружность.

(x,y) - координаты центра окружности или эллипса;

r - радиус круга или эллипса;

STEP - указывает, что (x,y) задаются как смещение относительно текущей позиции графического курсора;

n - цвет фигуры;

fi1,fi2 - начальный и конечный углы рисования фигуры. Их значения - в диапазоне от -2*PI до 2*PI радиан. При задании положительного угла рисуется лишь дуга, при задании отрицательного угла - та же дуга, но с отрезком, соединяющим точку дуги с центром.

k - коэффициент сжатия: отношение радиуса "y" к радиусу "x". По умолчанию его значение соответствует окружности, а его величина зависит от режима;

10.Оператор **PAINT [STEP](x,y)[,[n][,[m][,k$]]]** - закрашивает графические фигуры;

STEP - устанавливает режим отсчета координат относительно текущего положения курсора;

(x,y)- координаты начала закраски. Если точка внутри фигуры, то закрашивается сама фигура, если точка вне фигуры то закрашивается фон

n - атрибут цвета;

m - атрибут цвета для рамки;

k$ - параметр строкового типа, указывает шаблон цвета фона. Этот параметр используется для закраски уже окрашенной области экрана.

Следует отметить, что данный оператор правильно работает лишь в том случае, когда

- закрашиваемая область - замкнута;
- координаты (x,y) задают точку внутри окрашиваемой фигуры;
- окрашиваемая фигура ограничена рамкой цвета m.

Пример 11.5 Нарисовать на экране квадрат и полукруг; фигуры закрасить. (Использование операторов LINE, CIRCLE и PAINT).

```
CLS          ' Очистка экрана
SCREEN 9     ' Задание разрешения экрана 640 x 350 пикселей
REM  Задание коэффициента, выравнивающего x и y - координаты
a = 3 / 4

CLS          ' Очистка экрана
'Построение квадрата с заданной стороной n в центре экрана
n=100
LINE (320 - n/2, 175 - n/2*a)-(320 + n/2, 175 + n/2*a), 1, BF
REM   Построение полукруга
CIRCLE (160, 200), 75, 1, -.0001, -3.14
REM   Закраска полукруга
PAINT (160, 199), 2, 1
END
```

11. Оператор **VIEW PRINT [n to m**]- устанавливает границы текстового окна вывода.

n - числовое выражение: первая строка текстового вывода;

m - числовое выражение: последняя строка текстового вывода;

Если аргументы отсутствуют, окном является весь экран.

Пример 11.6 Сформировать на экране текстовое окно с бегущими строками. (Использование оператора VIEW PRINT).

```
CLS ' Очистка экрана
REM Установка границ окна: от 10-ой до 12-ой строки
VIEW PRINT 10 TO 15
FOR i% = 1 TO 200
PRINT USING " ###"; i% ;' Вывод чисел от 1 до 99
FOR j = 1 TO 500000: NEXT j ' Цикл задержки
NEXT i%
```

В результате выполнения программы на экране в строках с 10 по 12 побегут числа от 1 до 200.

12. Оператор **DRAW "выражение"** - рисует фигуру на экране в соответствии с содержимым параметра "выражение", заключенного в кавычки. Оператор позволяет достаточно просто рисовать сложные изогнутые траектории с помощью макроопределений, сведенных в таблицу. При этом команда перемещения рисует линию от точки последней ссылки к точке с координатами x,y.

Числовые аргументы в макрокомандах должны быть константами.

Команды перемещения	
Un	Вверх (на n пикселей)
Dn	Вниз
Ln	Влево
Rn	Вправо
En	По диагонали вверх и вправо
Fn	По диагонали вниз и вправо
Gn	По диагонали вниз и влево
Hn	По диагонали вверх и влево
Mx,y	Перемещение в точку с координатами (x,y)
B	Перемещение без отображения
N	Перемещение с возвратом в исходную точку

Команды вращения	
An Tan Sn	Установка угла поворота, кратного PI/2 рад (n=0, 1,2,3) Поворот на угол n (-360 <= n <= 360 град.). Положительный угол соответствует повороту против часовой стрелки. Установка масштабного коэффициента n - от 1 до 255. Число точек на экране в масштабной единице равно n/4. По умолчанию n=4.
Команды управления цветом	
Cp Pp,q	Установка цвета с номером n Выбор цвета закраски фигуры: p - номер цвета закраски, q - номер цвета границы

Пример 11.7 Нарисовать с помощью оператора DRAW "ключ".

```
SCREEN 11' Задание разрешения экрана 640 x 350
DRAW "M+25,+25 M+25,-25 M-25,-25 M-25,+25 M-60,+0"
DRAW "M+0,+20 M+10,-10 M+10,+10 M-0,-20"
END
```

В результате выполнения программы на экране появится графическое изображение ключа.

Положение изображения объекта на экране определяется координатами. Изменяя их, получим перемещение объекта. Для того чтобы создать иллюзию непрерывного движения, нужно следовать принципу мультипликации: стереть-нарисовать с достаточно большой частотой. Для улучшения качества изображения и устранения мерцания после рисования объекта необходимо его зафиксировать паузой. Это можно сделать, пустым циклом: FOR K=1 TO 100000: NEXT

Таким образом, для программирования движение объекта нужно выполнить следующую последовательность действий:

1. стереть прежнее изображение;
2. задать координаты нового изображения;
3. нарисовать новое изображение;
4. задержать пустым циклом;
5. повторить, начиная с первого пункта.

Пример 11.8 Построить на экране движущуюся фигуру, сформированную оператором DRAW. (Использование оператора DRAW).

```
SCREEN 9 'Задание разрешения экрана 640 x 350
FOR i = 1 TO 300 ' Цикл движения
FOR j = 1 TO 100000 : NEXT j 'Цикл задержки
REM Стирание старой фигуры (цвет - черный)
DRAW "c0e5u11f5r11g5d11h5l11"
REM Задание новой исходной точки:
PSET (173 + i, 177)
REM Формирование новой фигуры (цвет - красный)
DRAW "c12e5u11f5r11g5d11h5l11"
NEXT I
 END
```

В результате выполнения программы на экране отобразится фигура, движущаяся в горизонтальном направлении.

Возможно запрограммировать перемещение объекта по командам с клавиатуры.

Пример 11.9 Нарисовать фигуру робота, способного перемещаться по четырем направлениям в соответствии с нажатием клавиш управления.

```
SCREEN 12'640*480
x = 100: y = 240 ' задаем начальные координаты
50 PSET (x, y)
' рисуем фигуру:
 DRAW "c1r20d10r10d30l10u20d50l10u30d30l10u50d20l10u30r10u10"
100 A$ = INKEY$  ' запрашивается состояние клавиатуры
       IF A$ = "" THEN 100   ' при отсутствии нажатия запрос повторяется
' в противном случае объект стирается (рисуется черным на черном):
DRAW "c0r20d10r10d30l10u20d50l10u30d30l10u50d20l10u30r10u10"
'При нажатии клавиши "u" или "U" движение вверх
'При нажатии клавиши "d" или "D" движение вниз
'При нажатии клавиши "l" или "L движение влево
'При нажатии клавиши "r" или "R" движение вправо
IF A$ = "U" OR A$ = "u" AND y > 5 THEN y = y - 5
IF A$ = "D" OR A$ = "d" AND y < 450 THEN y = y + 5
IF A$ = "L" OR A$ = "l" AND x > 5 THEN x = x - 5
IF A$ = "R" OR A$ = "r" AND x < 600 THEN x = x + 5
IF A$ = "E" OR A$ = "e" THEN END
GOTO 50
END
```

13. Оператор **POINT (x,y)** определяет цвет точки с координатами x,y

Пример 11.10 Составить программу, увеличивающую изображение эллипса в заданное число раз.

```
CLS
INPUT m ' задаем увеличение в m раз
SCREEN 12
CIRCLE (5, 5), 5, 1, , , 1.5 ' в верхнем левом углу экрана рисуем эллипс
FOR j = 0 TO 10
FOR i = 0 TO 10
' Сканируем область верхнего левого угла экрана,
‘определяя цвет каждой точки:
k = POINT(i, j)
' Вычисляем координаты увеличенной фигуры:
x = i * m + 20
y = j * m + 20
' Вместо каждой точки с координатами (i,j) рисуем прямоугольник:
LINE (x, y)-(x + m - 1, y + m - 1), k, BF
NEXT
NEXT
```

14. Оператор **LOCATE n, m, i, l, k** - используется в текстовом режиме, перемещает курсор на указанную позицию экрана.

n - номер текстовой строки экрана; m - номер позиции в строке;

i - двоичное число делает курсор невидимым, при значении "0" и видимым при значении "1";

l, k - начало и конец курсора соответственно, эти параметры управляют размером курсора. по вертикали.

В качестве параметров могут быть переменные, арифметические выражения или числа. Все они должны быть целого типа. Любой из этих параметров может быть опущен.

Оператор PRINT, следующий за оператором LOCATE, выводит символы на экран, в позиции m строки n. Аналогично оператор INPUT, следующий за оператором LOCATE, запрашивает данные с позиции m, n.

Пример 11.11 Программа иллюстрирует работу оператора LOCATE.

```
CLS
x = 0
FOR i = 1 TO 7
' С каждым повторением в цикле курсор
'-  перемещается вниз по экрану на две строки
' - сдвигается вправо на две позиции
' - меняется размер от максимального к минимальному:
LOCATE 2 * i, 5 +  2* i, 1, i, 7
INPUT a  ' знак " ? " появляется на экране в  позиции "2 * i, 5 +  2* i "
PRINT ' пропуск строки
x = x + a  ' суммируются вводимые числа
NEXT
LOCATE , , 0  ' курсор выключается
FOR i = 1 TO 7
' С каждым повторением в цикле курсор
'-  перемещается вниз по экрану на две строки
' - сдвигается влево на две позиции
' - меняется размер от минимального к максимальному:
LOCATE 2 * i, 50 - i * 2, 1, 1, i
INPUT a   ' знак " ? " появляется на экране в  позиции "2 * i, 50 -  2* i "
PRINT
x = x + a
NEXT
LOCATE 16, 23, 0  ' невидимый курсор
PRINT x ' вывод значения суммы в 16-й строке, 23-й позиции
END
```

При выполнении программы Вы увидите, как перемещается курсор и меняется его размер.

Индивидуальное задание по лабораторной работе состоит из двух задач. Примеры типовых заданий по программированию в графическом режиме:

Задание 1

1. Построить столбчатую диаграмму (гистограмму) по заданным пяти исходным числам. Закрасить ее прямоугольники.
2. Построить совокупность кругов радиуса R со случайными координатами их центров.
3. Построить параллелепипед (в изометрии) с заданными координатами и раскрасить его грани в случайные цвета.

Задание 2

1. Написать программу, перемещающую по нажатию соответствующих клавиш исходный закрашенный круг по вертикали и горизонтали на один пиксель при каждом нажатии.
2. Создать программу выбора из меню трех типов графических фигур, которые после выбора последовательно (в строку) строятся на экране.
3. Сформировать "шахматную доску" с "фигурой" в виде квадрата, которая автоматически из заданного угла доски перемещается по краям доски ходами коня.

Лабораторное задание

1. Набрать и выполнить на ЭВМ примеры приведенные в описании. Проанализировать работу операторов, пользуясь отладочными режимами.

2. Выполнить на ЭВМ программу индивидуального задания результаты вывести на экран и в файл.

3. Написать отчет (краткий конспект и текст программ).

СОДЕРЖАНИЕ