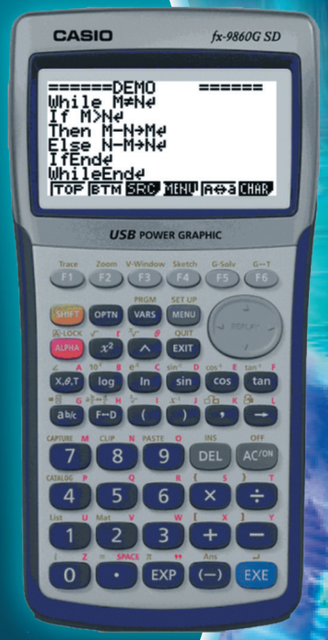
ISBN 978-5-91451-003-6

Востроknутов И.Е.
Помелова М.С.

Учимся программировать на графических калькуляторах CASIO FX-9860G

Учебное пособие по информатике и ИКТ для общеобразовательных учреждений



2008 г.

ВОСТРОКНУТОВ И.Е.,
ПОМЕЛОВА М.С.

УЧИМСЯ ПРОГРАММИРОВАТЬ НА ГРАФИЧЕСКИХ КАЛЬКУЛЯТОРАХ CASIO FX-9860G

Учебное пособие по информатике и ИКТ
для общеобразовательных учреждений

Москва
2008

УДК 004 (075.4)
ББК 74.261.63
В78

Вострокнутов И.Е., Помелова М.С.

В78 Учимся программировать на графических калькуляторах CASIO FX-9860G : учеб. пособие для общеобразовательных учебных заведений. — 2-е изд., испр. и доп. — М. : изд-во «Принтберри», 2008. — 60 с.

ISBN 978-5-91451-003-6

Книга представляет собой учебное пособие по реализации линии алгоритмизации и программирования на встроенном алгоритмическом языке программирования графических калькуляторов CASIO FX-9860G в школьном курсе информатики и ИКТ для общеобразовательных учреждений Российской Федерации в соответствии с действующим образовательным стандартом.

Приведено подробное описание основных алгоритмических конструкций встроенного языка программирования CASIO FX-9860G, рассмотренных на большом числе примеров. Содержится подборка задач для работы на уроке под руководством учителя и самостоятельной работы дома, что существенно облегчает практическое использование калькуляторов и данного пособия в обучении информатике.

Пособие предназначено для учащихся и учителей информатики и ИКТ общеобразовательных учреждений, а также для желающих самостоятельно освоить встроенный язык программирования графических калькуляторов CASIO FX-9860G.

УДК 004 (075.4)
ББК 74.261.63

ISBN 978-5-91451-003-6

ВВЕДЕНИЕ

Учебное пособие посвящено описанию языка программирования современных математических микрокомпьютеров компании Casio.

Компания Casio выпускает целую серию математических микрокомпьютеров, обладающих различными вычислительными возможностями, в том числе и возможностями программирования. Несмотря на все различия моделей калькуляторов, встроенный язык программирования обладает преемственностью, каждая последующая модель расширяет и дополняет его.

В данном пособии рассматриваются основные приемы программирования, характерные для программирования на языках высокого уровня типа BASIC и PASCAL. Мы хотели сохранить подходы к программированию на этих языках, которые достаточно полно освещены в различных учебных пособиях по BASIC и PASCAL. Но встроенный язык программирования математического микрокомпьютера обладает своими, только ему присущими особенностями, которые мы постарались учесть в этом пособии.

Учебный материал структурирован таким образом, что он позволяет учащимся изучить не только основные команды встроенного языка программирования, но и научиться основам программирования на современных языках, поскольку основные алгоритмические конструкции и структура команд совпадают с BASIC и PASCAL. Синтаксические особенности встроенного языка оговариваются.

Надеемся, что данное учебное пособие позволит не только глубже понять процесс вычислений на современных микрокомпьютерах, но и научиться программированию.

1. ЭТАПЫ РАЗРАБОТКИ ПРОГРАММ

Программирование — это процесс создания (разработки) программы. Разработка программ с использованием любого языка программирования, в том числе и встроенного языка программирования математического микрокомпьютера, происходит в несколько этапов:

1. **Постановка задачи.** На этом этапе подробно описывается, что должна представлять собой программа, какие будут использоваться входные данные, какой результат должен быть на выходе. (Заметим, что обязательно в программе должны присутствовать какие-то входные данные и на выходе результат в том или ином виде, в противном случае разработка программы теряет смысл.)

2. **Математическое или информационное моделирование.** На этом этапе создается математическая модель решаемой задачи, которая должна быть реализована на компьютере. Особое значение этот этап имеет при разработке сложных программ с использованием обработки данных и графических возможностей. В простейшем случае — это определение и описание данных в задаче, а также способов обработки этих данных.

3. **Разработка и выбор алгоритма.** Это определение последовательности действий, необходимых для достижения результата.

4. **Программирование.** Процесс написания и отладки программы на каком-либо языке программирования.

5. **Тестирование программы.** Этот этап используется при разработке сложных программ или программных комплексов, когда необходимо убедиться в том, что уже работающая программа выполняет именно то, что от нее требуется, и не выполняет то, что не требуется. Отладка заключается в тестировании программы на контрольных примерах.

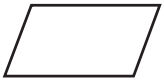
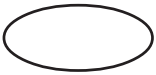
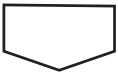
6. **Выполнение отлаженной программы и анализ результатов.** На этом этапе программист или пользователь вводит данные и анализирует полученный результат.

Алгоритм — это точное предписание, определяющее процесс перехода от исходных данных к результату в процессе выполнения разрабатываемой программы.

Начиная с 50-х годов XX века, для наглядного изображения алгоритмов, программисты стали использовать графические схемы, которые получили название блок-схем. Существует несколько видов блок-схем. Наибольшее распространение получили блок-схемы и структурограммы Насси-Шнейдермана.

Блок-схемы строятся по определенным правилам и включают в себя геометрические фигуры (блочные символы), соединенные между со-

Таблица 1. **Основные блочные символы**

Наименование	Обозначение	Функция
Процесс		Выполнение операции или группы операций, в результате которых изменяются значения данных или расположение данных
Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых условий
Модификация		Выполнение операций, меняющих команды или группы команд, меняющих программу
Предопределенный процесс		Использование ранее созданных и отдельно описанных алгоритмов
Ввод-вывод		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод)
Соединитель		Указание на наличие связи между прерванными линиями алгоритма обработки данных
Пуск-остановка		Начало, конец, прерывание данных или выполнения программы
Комментарий		Связь между элементом схемы и пояснением
Межстраничный соединитель		Указывает на наличие связи между разъединенными частями схемы, расположенными на разных страницах

бой стрелками (линиями), указывающими порядок выполнения операций. Все эти символы стандартизированы (ГОСТ 19.002-80 и ГОСТ 19.003-80, международные стандарты ISO 2636-73 или ISO 1038-73).

Наиболее часто используемые символы приведены в таблице 1.

Назначение блочных символов будет очевидным по мере изучения программирования и данного пособия.

2. ЯДРО МАТЕМАТИЧЕСКОГО МИКРОКОМПЬЮТЕРА

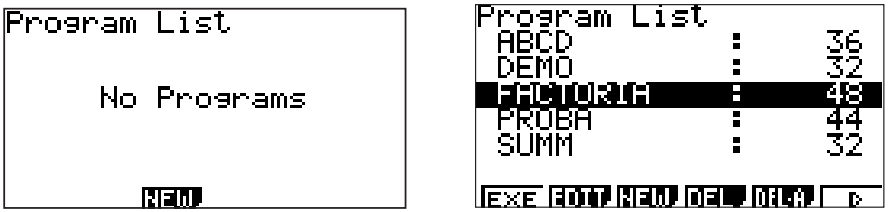
В основе любого языка программирования, в том числе математического микрокомпьютера, лежит ядро. Ядро программирования математического микрокомпьютера (калькулятора) — это минимальный набор средств, достаточный для написания сравнительно простых программ. С точки зрения программирования ядро можно рассматривать как набор основных правил, необходимых для написания программы, синтаксис языка, набор зарезервированных слов, стандартных процедур и функций.

Система программирования математического микрокомпьютера представляет собой объединение компилятора и инструментальной программной оболочкой, облегчающей пользователю разработку программ. В дальнейшем будем называть реализуемый компилятором встроенный язык программирования языком программирования CASIO (ЯПК), а разнообразные сервисные услуги, предоставляемые программной оболочкой, — средой программирования ЯПК.

2.1. Среда программирования ЯПК

Условно среду программирования можно разбить на две части: первая часть появляется при входе в режим программирования, вторая часть — режим редактирования программ.

Переход в режим программирования осуществляется из главного меню (MAIN MENU — вызов осуществляется нажатием клавиши [MENU]). С помощью клавиши [REPLAY] надо перевести курсор (выделить контрастно) на пиктограмму «программирование» (PRGM) и нажать клавишу [EXE]. Дисплей калькулятора приобретет вид, указанный на рис. 1*а*, если в памяти калькулятора нет программ, и



а *б*

Рис. 1. Режим программирования ЯПК

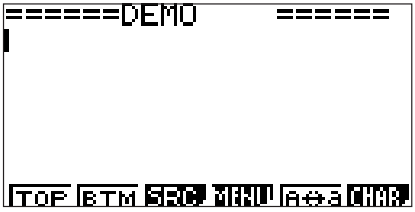


Рис. 2. Поле редактора математического микрокомпьютера

аналогичный указанному на рис. 1*б*, если в памяти калькулятора содержатся программы.

В данном режиме представлены программы, которые находятся в памяти калькулятора, но не являются принципиально необходимыми для обеспечения его работы. Каждая программа определяется именем и размером в байтах, который программа занимает в памяти. Некоторые программы могут быть помечены значком «*», указывающим, что они закрыты ключом.

Внизу под списком программ расположено меню программного режима.

Цепочка загрузки режима редактирования программы может иметь следующий вид:

MAIN MENU — PRGM — NEW — имя программы — [EXE]

После успешного вызова дисплей калькулятора приобретает вид, указанный на рис. 2.

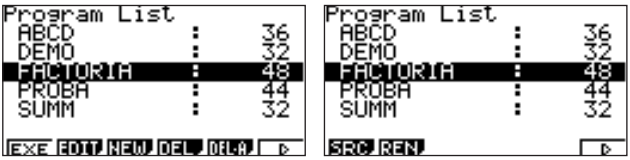
В верхней строке, очерченной двойной полосой, отображается имя программы. Нижняя строка содержит «встроенное меню», доступ к которому осуществляется с помощью нескольких управляющих клавиш (функциональные клавиши). Вся остальная часть принадлежит окну редактора, предназначенному для ввода и коррекции текста программы.

Для выхода из режима программирования следует нажать клавишу [MENU].

2.2. Функциональные клавиши

Функциональные клавиши используются для управления средой программирования калькулятора. Верхний ряд клавиатуры — 6 функциональных клавиш ([F1] — [F6]). Их текущее значение выводится в нижней строке экрана калькулятора над соответствующими клавишами. Если над клавишей ничего не выведено, значит, функциональ-

ная клавиша значения не имеет, и нажимать ее не следует. При входе в режим программирования меню функциональных клавиш принимает вид (с учетом продолжения при нажатии клавиши [F6]): EXE — [F1], EDIT — [F2], NEW — [F3], DEL — [F4], DEL-A — [F5], SRC — [F1], REN — [F2].



Назначение этих клавиш следующее:

EXE — исполнение программы, имя которой выделено цветом (с помощью клавиши [REPLAY]).

EDIT — редактирование уже имеющейся программы (открывается новый экран, на котором выведены ранее набранные команды программы; это тот же режим, что и режим создания новой программы после того, как ей было задано имя).

NEW — создание новой программы (открывается новый экран, в верхней строке которого предлагается ввести имя программы, далее — чистое поле для ввода команд программы. После ввода имени калькулятор автоматически переходит в режим редактирования программы).

DEL — удаление уже существующей программы (необходимо выделить программу).

DEL-A — выделение всех программ с возможностью удаления их всех.

SRC — поиск в программе заданной строки.

REN — переименование выделенной программы с предоставлением доступа к символам «'», «"», «—».

В режиме редактирования программ появляется новое меню функциональных клавиш (рис. 2).

TOP — переход к началу программы.

BTM — переход к концу программы.

SRC — поиск в программе заданной строки.

MENU — доступ к различным видам меню.

A↔a — переключение режима ввода символов из строчных в прописные.

CHAR — режим символов.

В режиме редактирования программ ввод служебных слов, с помощью которых записываются команды программ, осуществляется в

режиме команд. Для входа в него надо последовательно нажать клавиши [SHIFT] и [VARS] (PRGM). Меню функциональных клавиш изменится, список команд достаточно длинный, просмотр осуществляется с помощью клавиши [F6] (более подробное описание будет дано ниже).

2.3. Запуск и отладка программ

Для запуска программы необходимо выйти в режим программирования. Далее клавишей [REPLAY] выбрать программу и нажать [EXE].

Достаточно сложно осуществляется отладка программ математического микрокомпьютера, поскольку отсутствует развитая система пошаговой отладки программ. В случае возникновения ошибки в ходе исполнения программы вычисления приостанавливаются с выдачей сообщения об ошибке. Нажатием клавиши [EXIT] можно вывести листинг текущей программы, причем место ошибки в нем будет помечено курсором — мигающей горизонтальной или вертикальной чертой.

Особенностью калькулятора является автоматический вывод результата последней команды.

3. ЭЛЕМЕНТЫ ЯЗЫКА

3.1. Алфавит

Алфавит языка программирования калькулятора включает буквы, цифры, специальные символы и встроенные операторы. Буквы — это буквы латинского алфавита от A до Z, доступ к которым осуществляется последовательным нажатием клавиши [ALPHA] и соответствующей данной букве кнопки калькулятора (каждой кнопке соответствует буква алфавита, выделенная красным цветом). Калькулятор имеет большой набор специальных символов, доступ к которым осуществляется выбором режима ввода символов CHAR.

3.2. Встроенные операторы

В языке программирования калькулятора содержатся следующие встроенные операторы и символы:

?
◢
:
↵

COM (command) — команды управляющих структур:
If
Then
Else
IfEnd
For
To
Step
Next
While
WhileEnd
Do
LpWhile

CTL (control) — операции контроля и прерывания:
Prog
Return
Break
Stop

JUMP (jump) — операции переходов:
Lbl
Goto
⇒
Isz
Dsz

CLR (clear) — очистка:
ClrText
ClrGraph
ClrList
ClrMat
DrawStat

DISP (display) — вывод на дисплей:
DrawGraph
DrawDyna
DispF-Tbl
DrawFTG-Con
DrawFTG-Plt
DispR-Tbl
DrawWeb
DrawR-Con
DrawRΣ-Con
DrawR-Plt
DrawRΣ-Plt

REL (relative) — ввод операторов отношения:
=
≠
>
<
≥
≤

I/O (input/output) — команды ввода/вывода:
Locate
Getkey
Send
Receive
Send38k
Receive38k
OpenComport38k
CloseComport38k

Доступ к ним, а также ко всем командам и функциям, используемым в данном пособии, приведен в Приложении. Нет необходимости заучивать последовательность нажатия клавиш для доступа к тому или иному оператору, для этого предусмотрен каталог со всеми функциями калькулятора, который позволяет найти и ввести нужный оператор в программу. Доступ к каталогу осуществляется последовательным нажатием клавиш [SHIFT], [4] (CATALOG). Функции в каталоге представлены в виде списка в алфавитном порядке. Для быстрого поиска нужного оператора следует ввести его первую букву, затем выделить его и нажать клавишу [EXE].

При программировании используется ограниченный набор переменных и таблиц (массивов) с фиксированными именами. Эти переменные и списки являются глобальными, то есть общими для всех программ и подпрограмм. Имя переменной может состоять только из одной буквы, соответственно, переменных 28 (по числу букв, предусмотренных на клавиатуре калькулятора). Массивы только одномерные, всего их шесть (List1, ..., List6).

Размер программы ограничен объемом свободной памяти. Всего на программы в рассматриваемой модели математического микрокомпьютера выделено 64 Кбайта оперативной памяти (с учетом флеш-памяти в данной модели калькулятора имеется до 1,5 Мбайт). Для хранения в памяти символов, служебных слов, букв требуется по одному байту, для хранения чисел отводится два байта.

Для запуска программы необходимо выйти из режима редактирования программ клавишей [EXIT] в режим программирования, выделить имя программы клавишей [REPLAY] и нажать [EXE]. Исполняя программу, калькулятор выполняет все ее команды без остановки, прерываясь только при выполнении команд ввода и вывода. В первом случае он ждет введения числа, во втором — подтверждения, что программист видел выведенный результат, и программу можно исполнять дальше. Для этого служит [EXE].

Если в записи команды допущена ошибка или предписанное действие выполнить невозможно, калькулятор прекратит исполнение

программы, сообщит об ошибке и выведет на экран текст программы. При этом курсор будет указывать на команду, которую не удалось выполнить.

3.3. Особенности ввода текстовой информации

Вся текстовая информация вводится в режиме редактирования программ (рис. 2). Окно редактора имитирует длинный лист бумаги, ширина которого соответствует ширине дисплея калькулятора. Если строка не умещается по ширине окна, калькулятор разделяет ее и переносит оставшиеся символы на новую строку.

Воспользовавшись в процессе ввода и редактирования программы возможностями меню, описанными выше, можно смещать окно в начало или в конец текста, а также осуществлять поиск в программе заданной строки. Для перемещения курса по программе служит клавиша [REPLAY].

Редактирование осуществляется с помощью клавиши [DEL], которая стирает символ, расположенный слева от курсора. Редактор также может работать в режиме наложения новых символов на существующий старый текст. Для этого необходимо нажать [SHIFT], [DEL]. В этом режиме новый символ заменяет собой тот символ, на который указывает курсор.

Основной особенностью встроенного языка программирования математического микрокомпьютера является отсутствие клавиш с русскими буквами. Для этого в калькуляторе предусмотрено встроенное меню. Калькулятор имеет большой набор специальных символов, доступ к которым осуществляется нажатием CHAR ([F6]).

Ее нажатие приводит к появлению на экране большого списка символов: MATH, SYBL, АВГ, αβγ. Соответственно, к ним относятся:

Таблица символов АВГ содержит прописные буквы русского языка, таблица символов αβγ — строчные буквы русского языка. Перемещение по таблице символов осуществляется нажатием клавиши [REPLAY], причем для перехода из верхней строчки символов в нижнюю достаточно нажать REPLAY [▲]. Для выбора символа нужно переместить курсор на нужный символ и нажать [EXE]. Калькулятор вернется в режим редактирования программ, а в текст программы, в то место, где был расположен курсор, будет вставлен выбранный символ. Заметим, что для разделения слов используется пробел: [ALPHA], [·] (SPACE).

Недостатком ввода текста является необходимость постоянно обращаться к списку символов, чтобы ввести слово, поскольку данный режим позволяет осуществлять ввод лишь одного символа.

Калькулятор позволяет проводить следующие операции с текстом программы: выделение, копирование, удаление, вставка. Для выделения фрагмента текста необходимо установить курсор в начало или в конец фрагмента, который надо выделить, изменить форму курсора на  , нажав [SHIFT], [8] (CLIP), затем переместить курсор с помощью [REPLAY]. Это приведет к выделению фоном фрагмента текста, начиная с начального положения текста до его конечного положения. В случае сдвига курсора на следующую строку будет выделена вся строка целиком. При этом меняется меню функциональных клавиш: [F1] получает значение COPY (копировать), а [F2] — CUT (вырезать). Это классические команды редактирования. Если нужно копировать текст, необходимо выбрать COPY, при этом выделение фоном пропадет, а в памяти калькулятора будет сохранена его копия. Команда CUT удаляет выделенный фрагмент из текста, но сохраняет его в памяти. Сохраненный последним в памяти фрагмент текста можно вставить в любое место этой же или другой программы. Для этого необходимо с помощью [REPLAY] установить курсор в место вставки и нажать [SHIFT], [9] (PASTE). Фрагмент будет вставлен в указанное место, а исходный текст будет при этом раздвинут.

4. ОСНОВНЫЕ ОПЕРАТОРЫ ЯПК

4.1. Ввод-вывод информации в ЯПК

Оператор языка программирования графического калькулятора можно рассматривать как некий неделимый элемент программы, который позволяет выполнить определенные алгоритмические действия. Ядро языка программирования графического калькулятора содержит операторы, позволяющие реализовать основные алгоритмические конструкции.

Команды калькулятора могут размещаться в одну строку — в этом случае между ними помещается разделитель «**:**» (двоеточие). Окончание строки задается [EXE], которая на экране выводится в виде значка «↵». Если команда окончания строки не была введена, а место на экране в строке закончилось, то запись будет продолжена на следующей строке экрана, но все равно будет считаться одной строкой команд.

Решение даже самой простой задачи в программировании не обходится без операции ввода-вывода информации. В принципе, программирование теряет смысл, если не будет осуществляться ввод и вывод информации.

Простейшим оператором ввода является **оператор присваивания** (→). Служебное слово «→» представлено на клавиатуре специальной клавишей.

Команда присваивания имеет вид: «выражение» → «имя переменной» (или элемент списка).

Например:

1223,67→A

A+B→C

Для вывода информации на дисплей используются две записи команды вывода. Для вывода строки текста достаточно взять этот текст в кавычки. Например: "ПРИВЕТ"

Для вывода значений используется служебное слово ◢ (треугольник), размещаемое после имени переменной. Данный оператор приостанавливает вычисления и выводит надпись «-Display-». Такой вывод соответствует стилю вычислений на калькуляторе, однако при вы-

числениях по программе от этого стиля целесообразно отойти и позволить калькулятору автоматически сделать нужные вычисления и вывести их одновременно на экран дисплея. Для этого служит расширенный оператор вывода Locate (SHIFT , VARS (PRGM), F6 (▷), F4 (I/O), F1 (Lcte)), обеспечивающий, к тому же, вывод в любую заданную область дисплея, а не в очередную строку вывода. Этот оператор используется в следующей синтаксической форме записи: Locate N_столбца, N_строки, Выражение◢. Поскольку калькулятор автоматически выводит результат последней команды, поэтому можно в конце программы не ставить ◢, он автоматически выведет ответ. Это, в свою очередь, позволит исключить повторный вывод результата на дисплей. Хотя для того, чтобы у учащихся сложилось полноценное представление о программировании, целесообразно ставить ◢ и в конце программы, а для предотвращения повторного вывода значения вводить команду завершения программы:

"END" : Stop

Это необходимо также делать для того, чтобы показать успешное завершение программы.

Для ввода значений переменных с клавиатуры служит оператор «?» (подобный INPUT в BASIC). Его вид: ? → «имя переменной».

Пример 1.

Программа нахождения суммы двух чисел, введенных с клавиатуры.

```
"Введите 1 число" : ? →A↵
"Введите 2 число" : ? →B↵
A +B→C↵
" A +B =":C
```

или

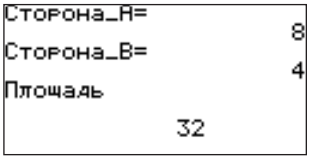
```
"Введите 1 число" : ? →A↵
"Введите 2 число" : ? →B↵
A +B→C↵
" A +B =":C◢
"END" : Stop
```

Выражение может быть числом, переменной или строкой вида «string». Значение выражения выводится в ту начальную позицию экрана, которое задано целочисленными номерами столбца и строки этой позиции. При этом позиция (1, 1) соответствует левому верхнему углу экрана, а позиция (21, 7) — нижнему правому углу. Таким образом, можно задавать N_столбца от 1 до 21 (всего 21 столбец) и N_строки от 1 до 7 (всего 7 строк).

Пример 2.

Программа, которая вычисляет площадь прямоугольника со сторонами 8 и 4 и выводит результат на экран.

```
8→A↵
4→B↵
A ×B→S↵
"Сторона A=" : A◢
"Сторона B=" : B◢
"Площадь =" : Locate 13, 7, S
```



4.2. Операции ЯПК

В язык программирования ЯПК заложены следующие арифметические операции:

+ — сложение;
- — вычитание;
× — умножение;
÷ — деление.

Операции отношения выполняют сравнение двух операндов и определяют истинность или ложность выражения. Запись соответствует математической.

Операции отношения

Операция	Название	Выражение
=	Равно	A=B
≠	Не равно	A≠B
>	Больше	A>B
<	Меньше	A<B
≥	Больше или равно	A≥B
≤	Меньше или равно	A≤B

Заметим, что в режиме программирования доступны все встроенные математические функции калькулятора (из всех режимов), а их всего более 240.

Основные математические функции ЯПК

Функция	Пояснение
X^2	возводит X в квадрат
X^Y	возводит X в степень Y
$\sqrt{X}$	извлекает квадратный корень из X
$\sqrt[3]{X}$	извлекает корень третей степени из X
$Y\sqrt[x]{X}$	извлекает корень степени Y из X

Функция	Пояснение
X!	вычисляет факториал X
Ran #	выводит случайное число от 0 до 1
Abs X	находит модуль X
sin X	находит sin числа X (X в радианах)
sin X°	находит sin числа X (X в градусах)
cos X	находит cos числа X (X в радианах)
cos X°	находит cos числа X (X в градусах)
tan X	находит tan числа X (X в радианах)
tan X°	находит tan числа X (X в градусах)
ln X	находит натуральный логарифм X
log X	находит десятичный логарифм X
e^X	возводит e в степень X
π	возвращает число π
Int X	находит целую часть X
Frac X	находит дробную часть X
Intg X	находит ближайшее целое, не превышающее X
d/dx($f(x)$,X)	находит производную $f(x)$ в точке X
$\int(f(x),a,b)$	находит определенный интеграл $f(x)$ на интервале (a, b)

Пример 3.
Выводит случайное число в интервале (0...1)

Ran # ◢
"END" : Stop

Пример 4.
Вывод случайных чисел в интервале (0...7)

Lbl 1 ↵
Ran # × 7 ◢
Goto 1

Поскольку данная программа рассчитана на бесконечное число повторений, для выхода из зацикливания следует нажать [AC], [EXIT].

Пример 5.
Вывод факториала

```
"Введите число N":?→N↵
"N!=" :N!◢
"END" : Stop
```

Задачи

1. Ввести число купленных тетрадей и карандашей. Вычислить стоимость покупки, если цена одной тетради 5 рублей, а карандаша — 2 рубля.

2. Ввести расстояние до дачи, количество бензина, которое автомашина потребляет в среднем на 100 км, и стоимость бензина. Вычислить стоимость поездки (туда и обратно).

3. Ввести длину дистанции, на которую бежал бегун, и его время. Вычислить среднюю скорость бегуна.

4. Ввести расстояние в километрах. Перевести расстояние в версты (1 верста — это 1066,8 м).

5. Ввести вес в килограммах. Перевести вес в фунты (1 фунт — это 405,9 грамм).

6. Ввести время, затраченное на выполнение домашнего задания, в часах и минутах. Перевести это время в минуты.

7. Ввести три целых числа. Найти их среднее геометрическое.

8. Ввести радиус окружности. Подсчитать длину окружности.

9. Ввести радиус окружности. Подсчитать площадь круга.

10. Ввести катеты a и b прямоугольного треугольника. Найти гипотенузу c.

11. Ввести длины двух сторон a и b произвольного треугольника и угол α между ними в градусах. Найти третью сторону c.

12. Ввести длины сторон произвольного треугольника a, b и c. Найти площадь треугольника.

13. Вычислить объем шара радиуса R. Длина R вводится с клавиатуры.

14. Ввести координаты двух точек на плоскости. Вычислить расстояние между ними.

15. Ввести длину грани куба. Вычислить площадь боковой поверхности и объем куба.

16. Ввести коэффициенты и значения правых частей системы уравнений. Найти ее решение. Вывести на экран вид системы уравнений и ответ.

17. Ввести длины катетов прямоугольного треугольника. Найти его гипотенузу и площадь.

18. Смешано V_1 литров воды температуры t_1 и V_2 литров воды температуры t_2. Найти объем и температуру образовавшейся смеси.

19. Найти площадь равнобедренной трапеции с основаниями a и b и углом при большем основании α. Значения a, b и α вводятся с клавиатуры.

20. Ввести длины сторон треугольника. Вычислить:

а) длины высот;

б) длины биссектрис;

в) длины медиан.

21. Ввести вещественные числа x и y. Вычислить расстояние от точки плоскости с координатами (x, y) до ближайшей границы квадрата с вершинами:

а) (-0,5; -0,5), (-0,5; 0,5), (0,5; 0,5), (0,5; -0,5);

б) (0; 0), (0; 1), (1; 1), (1; 0).

22. Ввести целые (либо вещественные) числа x_1, y_1, x_2, y_2, ..., x_n, y_n. Выяснить, найдутся ли среди точек с координатами $(x_1; y_1)$, (x_2, y_2), ..., (x_n, y_n) четыре такие, которые являются вершинами квадрата.

4.3. Условный оператор

Условный оператор позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действие. Условный оператор является средством ветвления программы.

Структура условного оператора имеет следующий вид:

If <условие> Then <оператор 1> Else <оператор 2> IfEnd

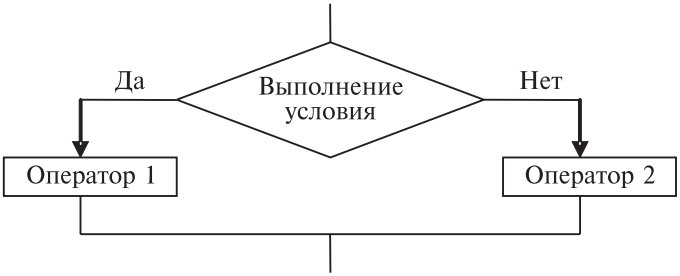
где If, Then, Else, IfEnd — встроенные операторы (если, то, иначе, конец):

<условие> — произвольное выражение логического типа;

<оператор 1>, <оператор 2> — любые операторы калькулятора.

Блок-схема условного оператора представлена на рис. 3.

Условный оператор работает следующим образом:



Сначала проверяется условное выражение <условие>. Если условие выполняется, то выполняется <оператор 1>, а <оператор 2> пропускается. Если условие не выполняется, то <оператор 1> пропускается, а выполняется <оператор 2>.

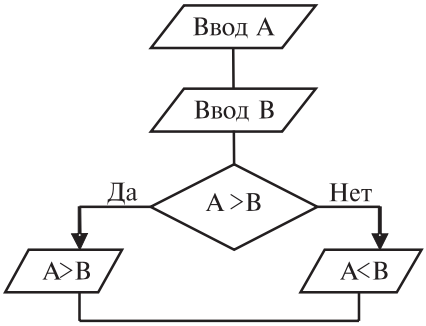


Рис. 3. Блок-схема условного оператора If — Then — Else — IfEnd

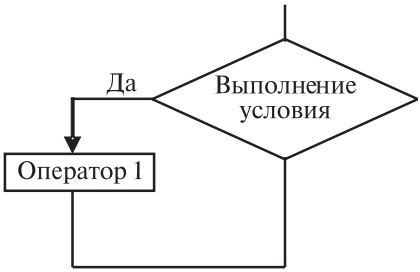


Рис. 4. Блок-схема условного оператора If — Then — IfEnd

Пример 6.

Ввести два различных числа. Определить минимальное и максимальное число.

```
"Введите 1 число":?→A↵
"Введите 2 число":?→B↵
If A>B↵
Then "Первое число больше второго"◢
Else "Второе число больше первого"◢
IfEnd↵
```

На практике часто используют укороченную схему условного оператора: If <условие> Then <оператор 1>

Пример 7.

```
"Введите 1 число":?→A↵
"Введите 2 число":?→B↵
If A>B↵
Then "Первое число больше второго"◢
IfEnd↵
If A<B↵
Then "Второе число больше первого"◢
IfEnd↵
If A=B↵
Then "Числа равны"◢
IfEnd
```

Если в зависимости от выполнения или невыполнения условия необходимо выполнить последовательность операторов, то они перечисляются:

```
If <условие> Then <оператор 1>↵
Then <оператор 2> ↵
Then <оператор 3> ↵
```

...
Then <оператор n>↵
IfEnd

«Оператор» в условном операторе может быть любым, в том числе и условным оператором. В таких случаях возникают конструкции с вложенными условиями типа:

If <условие1> Then <оператор 1>↵
Else If <условие2> Then <оператор 2> ↵
Else If <условие3> Then <оператор 3> IfEnd

В таких случаях удобно использовать составные условия.

Пример 8.
Ввести три числа. Определить максимальное число.

"Введите 1 число":?→A↵
"Введите 2 число":?→B↵
"Введите 3 число":?→C↵
If (A>B) And (A>C)↵
Then "Первое число максимум"◢
IfEnd↵
If (B>A) And (B>C)↵
Then "Второе число максимум"◢
IfEnd↵
If (C>B) And (C>A)↵
Then "Третье число максимум"◢
IfEnd↵
If (A=B) And (A=C)↵
Then "Числа равны"◢
IfEnd

Пример 9.
Ввести целые числа a, b, c. Выяснить, имеет ли уравнение $ax^2 + bx + c = 0$ действительные корни. Если имеет, то найти их и вывести на экран. В противном случае должно быть выведено сообщение, что корней нет.

"Введите число A":?→A↵
"Введите число B":?→B↵
"Введите число C":?→C↵
B^2-4A×C→D↵
If D<0↵
Then "Корней нет "↵
Else (-B+$\sqrt{}$ D)÷2A→Y↵
(-B-$\sqrt{}$ D)÷2A→O↵
"X1=": Y◢
"X2=": O◢

```
IfEnd↵
"END" : Stop
```

Задачи

23. Ввести три числа. Вывести их в порядке возрастания.

24. Ввести три числа. Вывести их в порядке убывания.

25. Ввести три числа. Удвоить эти числа, если $a > b > c$, и заменить их абсолютными значениями, если это не так.

26. Ввести два целых числа. Заменить первое число нулем, если оно меньше или равно второму, и оставить числа без изменения, если это не так.

27. Ввести два числа. Меньшее из этих чисел заменить их полусуммой, а большее — их удвоенным произведением.

28. Ввести целые числа a, b, c, d. Если a — наименьшее из них, то каждое число заменить на наибольшее; если $a > b > c > d$, то все числа заменить их квадратами; иначе числа оставить без изменения. Вывести полученные числа на экран.

29. Ввести числа x и y. Если оба числа отрицательны, то каждое заменить его модулем; если отрицательно только одно из них, то оба значения увеличить на 0,5; если оба значения неотрицательны и ни одно из них не принадлежит отрезку [0,5; 2,0], то оба значения уменьшить в 10 раз; в остальных случаях оставить числа без изменения. Вывести числа.

30. Ввести положительные числа a, b, c, x, y. Выяснить, пройдет ли кирпич с ребрами a, b, c в отверстие со сторонами x и y. Просовывать кирпич разрешается только так, чтобы каждое из ребер было параллельно или перпендикулярно каждой из сторон отверстия.

31. Ввести три числа x, y, z. Вывести те из них, которые принадлежат интервалу [0..10].

32. Ввести три числа x, y, z. Если их сумма меньше 10, то наименьшее из этих чисел заменить полусуммой двух других, в противном случае наибольшее из этих чисел заменить модулем полуразности двух других.

33. Ввести положительные числа x, y, z.

а) Выяснить, существует ли треугольник со сторонами, имеющими длины x, y, z.

б) Если треугольник существует, то ответить — является ли он остроугольным, тупоугольным или прямоугольным.

34. Ввести числа x_1, x_2, x_3, y_1, y_2, y_3. Определить, принадлежит ли начало координат треугольнику с вершинами, координаты которых (x_1, y_1), (x_2, y_2), (x_3, y_3).

35. Составить программу определения большей площади из двух фигур: круга и квадрата. Сторона квадрата равна a, а радиус круга r. Значения a и r вводятся с клавиатуры. Вывести на экран значение площади большей фигуры.

36. Ввести четыре вещественных положительных числа *a*, *b*, *c* и *d*. Определить, можно ли построить четырехугольник со сторонами, имеющими такие длины.

4.4. Операторы повторения

Для создания циклов с заданным числом повторений используется счетный оператор цикла For, который имеет следующую структуру:

For <Нач. Знач.> → <Парам. Цик.> To <Кон. Знач.>
{R} <оператор {R}> Next

Здесь For, To, Next — встроенные операторы (для, до, выполнить);

<Парам. Цик.> — параметр цикла — переменная любого порядкового типа;

<Нач. Знач.> — начальное значение — выражение того же типа;

<Кон. Знач.> — конечное значение — выражение того же типа;

<оператор {R}> — любой оператор калькулятора.

Блок-схема счетного оператора циклов For представлена на рис. 5.

Оператор For работает следующим образом. Сначала вычисляется выражение <Нач. Знач.> и осуществляется присваивание параметру цикла начального значения. После этого циклически повторяется:

проверка условия <Парам. Цик.> ≤ <Кон. Знач.>;

если условие выполнено, оператор For завершает свою работу.

Пример 10.

Программа, которая выводит на экран последовательность целых чисел от 1 до 10.

```
For 1→I To 10 ↵
I◢
Next
```

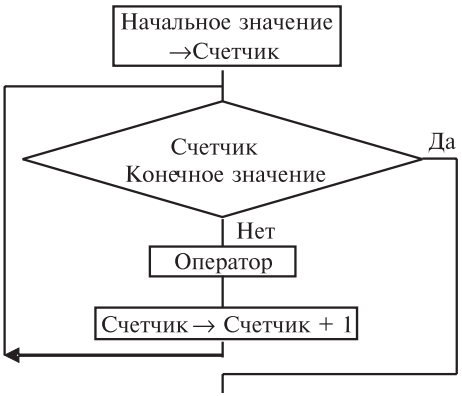


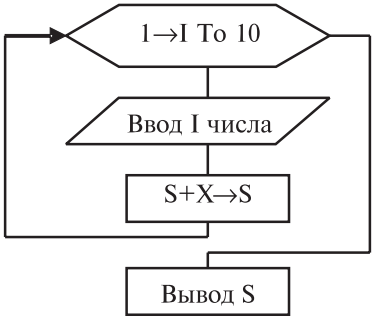
Рис. 5. Блок-схема выполнения счетного оператора циклов For

Следует отметить, что оператор For может выполнять циклически несколько операторов.

- выполнение оператора <оператор {R}>;
- наращивание переменной <Парам. Цик.> на единицу.

Пример 11.

Программа, которая вычисляет сумму десяти чисел, последовательно введенных с клавиатуры.




```
0 → S ↵
For 1 → I To 10 ↵
"Введите число" : ? → X ↵
S + X → S ↵
Next ↵
S◢
"END" : Stop
```

Следует отметить, что условие, управляющее работой оператора For, проверяется перед выполнением оператора. Если условие не выполняется в самом начале работы оператора, то он не выполняется ни разу. Следующее обстоятельство — шаг наращивания параметра строго постоянен и равен «+1».

Если необходимо, чтобы изменения происходили с произвольным шагом, то надо применить более общую конструкцию цикла:

For <Нач. Знач.> → <Парам. Цик.> To <Кон. Знач.>
Step <Разм. шага> {R} <оператор {R}> Next

Здесь <Разм. шага> — изменение переменной — может быть как целым, так и дробным числом.

В данном случае возможен цикл с обратным порядком, уменьшением значения переменной.

Пример 12.

Программа, которая выводит на экран последовательность целых чисел от 10 до 1.

```
For 10→I To 1 Step -1 ↵
I◢
Next ↵
"END" : Stop
```

Пример 13.

Программа, которая выводит на экран числовую последовательность от 1 до 5 с шагом 0,01.

```
For 1→I To 5 Step 0.01 ↵
I◢
Next ↵
"END" : Stop
```

Пример 14.

Программа, которая находит факториал.

```
"N=" : ? →N ↵
1 → F ↵
For 1→I To N ↵
F × I → F ↵
Next ↵
"N!=" : F◢
"END" : Stop
```

Оператор цикла While с предпроверкой условия:

While <условие> {R} <оператор {R} > WhileEnd

Здесь While, WhileEnd — встроенные операторы (пока [выполняется условие], конец);

<условие> — выражение логического типа;

<оператор {R}> — произвольный оператор калькулятора.

Блок-схема оператора цикла While представлена на рис. 6.

Оператор начинает свою работу с вычисления и проверки условия. Если условие выполняется, то выполняется <оператор>, после

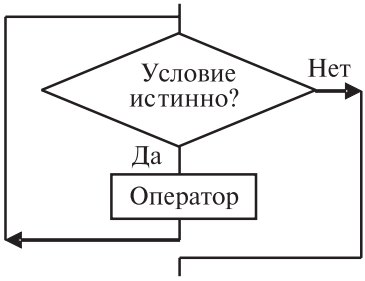


Рис. 6. Блок-схема оператора цикла While

чего вычисление выражения <условие> и его проверка повторяются. Если условие не выполняется, то оператор цикла While прекращает свою работу.

Рассмотрим пример 13, реализованный с помощью оператора цикла While.

```
1 → I ↵
While I < 5 ↵
I + 0.01 → I ↵
I◢
WhileEnd ↵
"END" : Stop
```

Пример 15.

Рассмотрим следующую задачу. Даны целые числа *n* и *m*. Найти наибольший делитель этих чисел. Для решения данной задачи удобно использовать алгоритм Евклида. Идея этого алгоритма основана на том свойстве, что если $m > n$, то НОД(m, n) = НОД($|m-n|$, n). Иначе говоря, НОД двух натуральных чисел равен НОД модуля их разности и меньшего числа. Тогда, если $m = n$, то НОД(m, m) = m.

Например: НОД(15, 6) = НОД(9, 6) = НОД(3, 6) = НОД(3, 3) = 3

```
"Введите M" : ? →M↵
"Введите N" : ? →N↵
While M≠N↵
If M>N↵
Then M-N→M↵
Else N-M→N↵
IfEnd↵
WhileEnd↵
M◢
"END" : Stop
```

Задачи

37. Ввести 10 целых чисел. Найти их среднее арифметическое.

38. Ввести 10 целых чисел. Найти их среднее геометрическое.

39. Ввести *n* чисел. Подсчитать сумму положительных и сумму отрицательных чисел. Найти общую сумму.

40. Ввести *n* чисел. Подсчитать число положительных и отрицательных чисел.

41. Ввести *n* чисел. Определить минимальное и максимальное число.

42. Ввести *n* чисел. Определить четные и нечетные числа.

43. Ввести *n* чисел. Определить числа, одновременно кратные 3 и 5.

44. Ввести *n* чисел. Определить числа, которые при делении на 5 дают остаток 2.

45. Вывести на экран ряд чисел от 10 до 30 с шагом 0, 01.

46. Дано целое число n. Вычислить:

а) 2^n;

б) $n!$;

в) $\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)...\left(1+\frac{1}{n^2}\right)$;

г) $\underbrace{\sqrt{2+\sqrt{2+...\sqrt{2}}}}_{n-корней}$;

д) $\sqrt{3+\sqrt{6+\sqrt{9+...+\sqrt{3(n+1)+3\sqrt{n}}}}}$.

47. Дано целое число a, целое число n. Вычислить:

$$\underbrace{((...((}_{\text{n скобок}} x+a)^2+a)^2+...+a)^2+a.$$

48. Дано вещественное число a. Найти:

а) среди чисел $1,\ 1+\frac{1}{2},\ 1+\frac{1}{2}+\frac{1}{3},\ ...$ первое, большее a;

б) такое наименьшее n, что $1,\ 1+\frac{1}{2},\ 1+\frac{1}{2}+\frac{1}{3},\ ...,\ 1+\frac{1}{2}+...+\frac{1}{n}$ первое, большее a;

в) такое наименьшее n, что $1+\frac{1}{2}+...+\frac{1}{n}>a$.

49. Дано вещественное число n.

а) Сколько цифр в числе n?

б) Чему равна сумма его цифр?

в) Найти первую цифру числа n.

50. Даны вещественные числа m и n. Получить сумму m последних цифр числа n.

а) Выяснить, входит ли цифра 3 в запись числа n^2.

б) Поменять порядок цифр числа n на обратный.

в) Переставить первую и последнюю цифру числа.

г) Приписать по единице в начало и в конец записи числа n.

51. Пусть $x_1=y_1=1$, $x_i=0{,}3x_{i-1}$, $y_i=x_{i-1}+y_{i-1}$, $i=2, 3, ...$ Дано целое n. Найти $\sum_{i=1}^{n}\frac{x_i}{1+|y_i|}$.

52. Дано целое число n. Получить наименьшее число вида 2^r, превосходящее n.

53. Дано целое число n. Вычислить $1\cdot 2 + 2\cdot 3 + 3\cdot 4 + \ldots + n(n+1)$.

54. Вычислить $\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{5+\cfrac{1}{\ldots}}}}$.

$$\cfrac{}{101+\cfrac{1}{103}}$$

55. Дано целое число $x \neq 0$. Вычислить $\cfrac{x}{x^2+\cfrac{2}{x^2+\cfrac{4}{x^2+\cfrac{8}{\ldots}}}}$.

$$\cfrac{}{x^2+\cfrac{256}{x^2}}$$

56. Даны целые числа n, k ($n \geq k \geq 0$). Вычислить $\dfrac{n(n-1)\ldots[n-k+1]}{k!}$.

57. Вычислить а) $\sum\limits_{i=1}^{100}\dfrac{1}{i^2}$; б) $\sum\limits_{i=1}^{50}\dfrac{1}{i^3}$; в) $\sum\limits_{i=1}^{10}\dfrac{1}{i!}$; г) $\sum\limits_{i=1}^{128}\dfrac{1}{(2i)^2}$; д) $\prod\limits_{i=1}^{52}\dfrac{i^2}{i^2+2i+3}$; е) $\prod\limits_{i=1}^{10}\left(2+\dfrac{1}{i!}\right)$; ж) $\prod\limits_{i=2}^{100}\dfrac{i+1}{i+2}$; з) $\prod\limits_{i=2}^{10}\left(1+\dfrac{1}{i!}\right)^2$.

(Выражение $\prod\limits_{i=1}^{52}\dfrac{i^2}{i^2+2i+3}$ — это краткая запись произведения $\dfrac{1^2}{1^2+2\cdot 1+3}\cdot\dfrac{2^2}{2^2+2\cdot 2+3}\cdot\ldots\cdot\dfrac{52^2}{52^2+2\cdot 52+3}$.)

58. Дано целое число n. Вычислить: а) $\sum\limits_{k=1}^{n}\dfrac{1}{k}$; б) $\sum\limits_{k=1}^{n}\dfrac{1}{k^5}$; в) $\sum\limits_{k=1}^{n}\dfrac{1}{(2k+1)^2}$; г) $\sum\limits_{k=1}^{n}\dfrac{(-1)^k}{(2k+1)k}$; д) $\sum\limits_{k=1}^{n}\dfrac{(-1)^{k+1}}{k(k+1)}$; е) $\sum\limits_{k=1}^{n}\dfrac{(-1)^k(k+1)}{k!}$; ж) $\sum\limits_{k=1}^{n}\dfrac{k!}{\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{k+1}}$.

59. Дано целое число n. вычислить произведение первых n сомножителей: а) $\dfrac{1}{2}\cdot\dfrac{3}{4}\cdot\dfrac{5}{6}\cdot\ldots\cdot\dfrac{n-1}{n}$; б) $\dfrac{1}{1}\cdot\dfrac{3}{2}\cdot\dfrac{5}{3}\cdot\ldots\cdot\dfrac{n}{n-1}$.

В программах достаточно часто используют конструкции, в которых в теле циклических операторов могут находиться другие циклические операторы. Такие внутренние циклы называют вложенными.

```
For 1→I To N ↵
For 1→J To N ↵
...
Next ↵
...
Next ↵
```

Здесь J — внутренний цикл, I — внешний цикл, сначала закрывается внутренний цикл, затем — внешний.

Подобные конструкции работают следующим образом. В начальной момент переменной I присваивается значение 1, переменной J — значение 1. Затем выполняется тело цикла. После выполнения тела цикла переменная J увеличивает значение на 1, а переменная I остается равной 1. Затем опять выполняется тело цикла. Это повторяется до тех пор, пока значение переменной J не превысит конечного значения. После этого значение I увеличится на 1, а J станет равным 1. ... Таким образом, чтобы внешний цикл увеличил свое значение на 1, необходимо, чтобы выполнился весь внутренний цикл.

Чаще всего в программировании вложенные циклы применяются в задачах для возведения в произвольную степень. ЯПК является более совершенным языком, нежели BASIC и PASCAL, и позволяет возводить и извлекать корень любой степени, находить факториал и т.п. Поэтому целесообразно рассмотреть данные задачи с применением двойного цикла, а затем с применением стандартных функций калькулятора.

Пример 16.

Ввести целое число *N*. Вычислить сумму $\left(\frac{1}{1}\right)^N + \left(\frac{1}{2}\right)^N + ... + \left(\frac{1}{N}\right)^N$.

Программа с применением вложенных циклов	Программа с применением функции возведения в произвольную степень
0 → S ↵ "Введите число членов последовательности" : ? →N ↵ For 1→I To N ↵ 1÷I→ A ↵ A→ P ↵ For 2→ J To N ↵ P× A→ P ↵ Next ↵ S+P→ S ↵ Next ↵ "Сумма степеней =" : S◢ "END" : Stop	0 → S ↵ "Введите число членов последовательности" : ? →N ↵ For 1→I To N ↵ (1÷I)^N→ A ↵ S+A→ S ↵ Next ↵ "Сумма степеней =" : S◢ "END" : Stop

Задачи

60. Ввести числа a_1, ..., a_{10}. Вычислить $a_1 + a_2^2 + ... + a_{10}^{10}$.

61. Ввести числа a_1, ..., a_{10}. Получить последовательность b_1, ..., b_{10}, в которой

$$b_1 = a_1 + a_2 + ... + a_{10}$$
$$b_2 = a_1^2 + a_2^2 + ... + a_{10}^2$$
$$... \quad ... \quad ...$$
$$b_{10} = a_1^{10} + a_2^{10} + ... + a_{10}^{10}$$

62. Ввести целое число n. Вычислить последовательность:

а) $\left(\frac{1}{1}\right)^n$, $\left(\frac{1}{2}\right)^n$, ..., $\left(\frac{1}{n}\right)^n$;

б) $\left(\frac{1}{1}\right)^1$, $\left(\frac{1}{2}\right)^2$, ..., $\left(\frac{1}{n}\right)^n$;

в) $\frac{1}{1^1}$, $\frac{1}{2^2}$, ..., $\frac{1}{n^n}$;

г) $\left(\frac{1}{1}\right)^n$, $\left(\frac{1}{2}\right)^{n-1}$, ..., $\left(\frac{1}{n}\right)^1$;

д) $\frac{1}{1^n}$, $\frac{1}{2^{n-1}}$, ..., $\frac{1}{n^1}$.

63. Ввести целое число n. Вычислить последовательность:

а) $\left(1+\frac{1}{1^n}\right)$, $\left(1+\frac{1}{2^n}\right)$, ..., $\left(1+\frac{1}{n^n}\right)$;

б) $\left(1+\frac{1}{1^1}\right)$, $\left(1+\frac{1}{2^2}\right)$, ..., $\left(1+\frac{1}{n^n}\right)$;

в) $\left(1+\frac{1}{1}\right)^n$, $\left(1+\frac{1}{2}\right)^n$, ..., $\left(1+\frac{1}{n}\right)^n$;

г) $\left(1+\frac{1}{1}\right)^1$, $\left(1+\frac{1}{2}\right)^2$, ..., $\left(1+\frac{1}{n}\right)^n$;

д) $\left(1+\frac{1}{1}\right)^n$, $\left(1+\frac{1}{2}\right)^{n-1}$, ..., $\left(1+\frac{1}{n}\right)^1$.

64. Ввести целое число n, вещественные числа a_1, ..., a_n. Вычислить:

а) $a_1^n + a_2^n + ... + a_n^n$;

б) $a_1^1 + a_2^2 + ... + a_n^n$;

в) $a_1^1 - a_2^2 + ... + (-1)^{n+1} a_n^n$;

г) $a_1 + a_2(a_2 - 1) + ... + a_n(a_n - 1) ... (a_n - n + 1)$.

65. Ввести числа m и n. Получить все меньшие n числа, квадрат суммы цифр которых равен m.

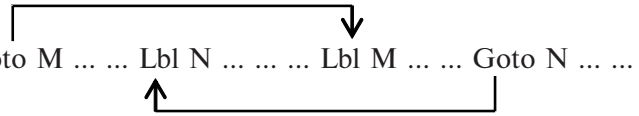
66. Ввести целое число n. Определить, можно ли n представить в виде суммы квадратов трех чисел. Если можно, то:

а) указать тройку x, y, z таких целых чисел, что $x^2 + y^2 + z^2 = n$;

б) указать все тройки x, y, z таких целых чисел, что $x^2 + y^2 + z^2 = n$.

4.5. Оператор безусловного перехода

Несмотря на то, что язык калькулятора является структурным языком программирования, в операторах ветвления может использоваться оператор безусловного перехода Goto, имеющий следующую структуру:



<Lbl M> — метка, где M — любая цифра или буква латинского алфавита;

<Goto M> — оператор, обеспечивающий безусловный переход к метке Lbl M.

Если в программе есть запись вида:

метка: оператор

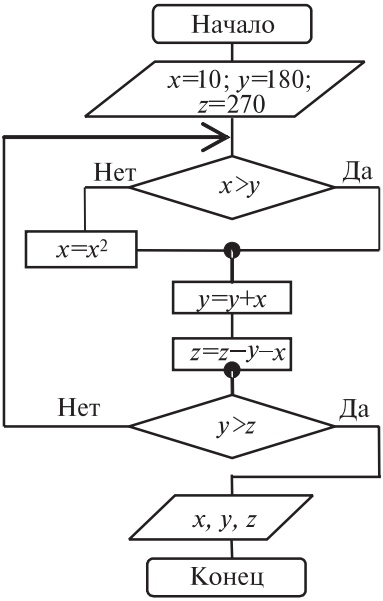
то после выполнения оператора безусловного перехода выполняется оператор с соответствующей меткой, а далее выполнение программы продолжается с этого места.

Следует применять данный оператор при создании простых программ, чтобы избежать путаницы при создании безусловных переходов.

В настоящее время применение оператора безусловного перехода считается плохим тоном в программировании и часто говорит о низкой квалификации программиста, но, тем не менее, в языках программирования, подобных ЯПК, оператор безусловного перехода может успешно применяться для решения задач, подобных примеру 17. Здесь удобно применять циклический оператор с постпроверкой условия, но сначала следует более наглядно рассмотреть решение с оператором ветвления.

Пример 17.

Вычислить значение переменных x, y, z в соответствии с алгоритмом сначала с оператором ветвления, затем с циклическим оператором:



Следует обратить внимание на то, что данная задача решается по обратному условию.

Решение с использованием оператора ветвления:

```
10→X↵
180→Y↵
270→Z↵
Lbl M : If X<Y↵
Then X²→X ↵
IfEnd ↵
Y+X→Y ↵
Z-Y-X→ Z ↵
If Y<Z↵
Then Goto M ↵
IfEnd↵
X◢
Y◢
Z◢
"END" : Stop
```

Решение с использованием циклического оператора:

```
10→X ↵
180→Y ↵
270→Z ↵
While Y<Z ↵
If X<Y ↵
Then X²→X ↵
IfEnd ↵
X+Y→Y ↵
Z-Y-X→Z ↵
WhileEnd ↵
X◢
Y◢
Z◢
"END" : Stop
```

Ответ: $x = 100$, $y = 280$, $z = -110$.

Задачи

67. Ввести n вещественных чисел. Определить, сколько среди них отрицательных чисел.

68. Ввести n вещественных чисел. Определить, сколько среди них положительных чисел.

69. Ввести n вещественных чисел. Найти их сумму.

70. Ввести n вещественных чисел. Определить сумму чисел, стоящих на четных местах в этой последовательности.

71. Ввести n вещественных чисел. Определить сумму положительных чисел.

72. Ввести n вещественных чисел. Определить сумму отрицательных чисел.

73. Ввести целое k от 1 до 180. Определить, какая цифра находится в k-ой позиции последовательности 10111213...9899, в которой выписаны подряд все двузначные числа.

74. Вычислить c — наибольший общий делитель натуральных чисел a и b.

75. Для заданного числа a найти корень уравнения $f(x) = 0$, где

$$f(x) = \begin{cases} 2ax + |a-1|, & a < 0, \\ \dfrac{x}{\sqrt{1+a^2}} - 1, & a \notin [0, \infty]. \end{cases}$$

76. Ввести число x. Напечатать в порядке возрастания числа x, $(1 + |x|)$ и $(1 + x^2)$.

5. МАССИВЫ

Массив объединяет элементы одного типа данных. Более формально массив можно определить как одномерную (последовательную) упорядоченную совокупность элементов некоторого типа, которые адресуются с помощью индекса. В качестве иллюстрации можно представить себе шкаф, имеющий множество пронумерованных ящиков. Доступ к содержимому конкретного ящика (элемента данных) осуществляется после выбора ящика по его номеру (индексу).

В литературе, наряду с термином «массив», часто можно встретить термин «матрица», «таблица» или «вектор». Суть всех этих терминов одна и та же.

Отличительной особенностью языка программирования калькулятора является то, что в нем предусмотрено только 6 одномерных массивов. В математике и информатике массив называется одномерным, если для получения доступа к его элементу достаточно одной индексной переменной. Массивы при этом различаются по номерам: List 1, List 2, ..., List 6.

В отличие от других языков программирования в калькуляторе массивы могут обрабатываться как целиком, так и отдельно, каждый элемент.

Целиком массивы обрабатываются в том случае, если используются встроенные математические функции калькулятора (например, сортировка функций SortA, описанная ниже).

5.1. Доступ к элементам массива

Язык программирования калькулятора позволяет применять стандартные алгоритмы рассмотрения одномерных массивов, т.е. получая доступ к каждому элементу-ячейке. Это выполняется путем указания индекса в квадратных скобках. Например, с помощью оператора

```
15 →List 1 [ 3 ] ↵
```

элементу массива 1 с индексом 3 (ячейке с номером 3) присваивается значение 15. Наряду с конкретным значением в качестве индекса может использоваться переменная, например, при поэлементарной обработке массива в рамках цикла For ... To ... Next. Так с помощью фрагмента

```
For 1→I To 7 ↵
0→List 1 [ I ] ↵
Next
```

всем элементам массива 1 присваивается значение 0. Причем нумерация элементов массива всегда начинается с 1.

Для заполнения массивов используются следующие конструкции:

```
For 1→I To 4 ↵
?→List 1 [ I ] ↵
Next
```

или

```
"Введите длину массива" : ? →N ↵
For 1→I To N ↵
"Введите элемент" : I◢
?→List 1 [ I ] ↵
Next
```

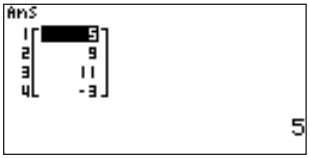
Для вывода массива не следует вновь вводить цикл. Достаточно указать номер массива: List 1◢.

Выводят массив на дисплей следующим образом:

Пример. Программа, которая сначала обеспечивает ввод четырех чисел в массив, затем выводит его на дисплей.

```
For 1→I To 4 ↵
"Введите элемент" : I◢
?→List 1 [ I ] ↵
Next ↵
List 1◢
```

На дисплее массив выводится в следующем виде:



Пример 18.

Программа, которая сначала обеспечивает ввод десяти чисел в одномерный массив, затем складывает их и выводит результат на экран.

```
For 1→I To 10 ↵
"Введите элемент" : I◢
?→List 2 [ I ] ↵
Next ↵
0 →S ↵
For 1→I To 10 ↵
S + List 2 [ I ] →S ↵
Next ↵
"Сумма 10 элементов =" : S◢
"END" : Stop
```

Пример 19.

Программа, которая обеспечивает ввод с клавиатуры и сохранение в одномерном массиве 15 чисел. Затем формирует запрос на ввод еще одного числа. После ввода данного числа программа проверяет элементы массива на наличие в них числа, равного последнему введенному, и, если такое число есть, выдает соответствующее сообщение на экран.

```
0 →R ↵
For 1→I To 15 ↵
"Введите элемент" : I◢
?→List 2 [ I ] ↵
Next ↵
"Введите число" : ? →S ↵
For 1→I To 15 ↵
If S= List 2 [ I ] ↵
Then R+1 →R ↵
"Обнаружено" : R◢
"-е вхождение числа" : S◢
"в позиции" : I◢
IfEnd ↵
Next ↵
If R≠0 ↵
Then " Итого число" : S◢
"встречается" : R◢
"раз" ↵
Else "Число" : S◢
"не встречается ни разу" ↵
IfEnd
```

Пример 20.

Ввести два массива из *n* — элементов целых чисел. Найти их сумму и разность.

```
ClrList↵
"Введите длину массивов 1 и 2 " : ? →N ↵
For 1→I To N ↵
"Введите элемент массива 1" : I◢
?→List 1 [ I ] ↵
Next ↵
For 1→J To N ↵
"Введите элемент массива 2" : J◢
?→List 2 [ J ] ↵
Next ↵
List 1◢
List 2◢
```

```
List 1 + List 2→List 3 ↵
List 3◢
List 1 — List 2→List 4 ↵
List 4◢
"END" : Stop
```

Пример 21.

Ввести массив из *n* — элементов целых чисел. Определить позицию и величину минимального элемента.

```
ClrList↵
"Введите длину массива " : ? →N ↵
For 1→I To N ↵
"Введите элемент массива " : I◢
?→List 2 [ I ] ↵
Next ↵
1→J: List 2 [ 1 ] →S ↵
For 2→I To N ↵
If List 2 [ I ]<S ↵
Then List 2 [ I ] →S: I→J ↵
IfEnd ↵
Next ↵
"минимальный элемент =": S◢
"номер элемента =": J◢
"END" : Stop
```

или

```
ClrList↵
"Введите длину массива " : ? →N ↵
For 1→I To N ↵
"Введите элемент массива " : I◢
?→List 2 [ I ] ↵
Next ↵
"минимальный элемент =":Min(List 2) ◢
```

Задачи

77. Ввести массив из *n* элементов целых чисел. Определить элементы, которые:

а) являются нечетными числами;

б) кратны 3 и 5;

в) являются удвоенными нечетными числами.

78. Ввести массив из *n* элементов целых чисел. Определить элементы, которые:

а) при делении на 5 дают остаток 2;

б) при делении на 7 дают остаток 3;

в) при делении на 3 дают остаток 1.

79. Ввести массив из n элементов целых чисел. Определить элементы, удовлетворяющие условию:

а) $A[i] < i$;

б) $A[i] < i^2$;

в) $A[i] > 10\text{-}i$.

80. Ввести массив из n элементов целых чисел. Выяснить, верно ли, что:

а) сумма первой половины матрицы больше суммы второй половины матрицы;

б) максимальный элемент матрицы находится в первой половине;

в) матрица симметрична.

81. Ввести массив из n элементов целых чисел. Заменить все отрицательные числа их модулями.

82. Ввести массив из n элементов целых чисел. Определить повторяющиеся числа.

83. Ввести массив из n элементов целых чисел. Подсчитать число соседств положительных и отрицательных чисел.

84. Ввести массив из n элементов целых чисел. Определить позицию и величину минимального и максимального элемента.

85. Ввести массив из n элементов целых чисел. Заменить все большие 7 элементы массива числом 7. Подсчитать число таких элементов.

86. Ввести массив из n элементов целых чисел. Выяснить, какое число встречается в массиве раньше — положительное или отрицательное. Если все элементы массива равны нулю, то сообщить об этом.

5.2. Сортировка массивов

Все методы сортировки можно разделить на две большие группы:

— прямые методы сортировки;

— улучшенные методы сортировки.

Прямые методы сортировки по принципу, лежащему в основе метода, в свою очередь разделяются на три подгруппы:

1) сортировка вставкой (включением);

2) сортировка выбором (выделением);

3) сортировка обменом («пузырьковая» сортировка).

Улучшенные методы сортировки основываются на тех же принципах, что и прямые, но используют некоторые оригинальные идеи для ускорения процесса сортировки. Прямые методы на практике используются довольно редко, так как имеют относительно низкое быстродействие. Однако они хорошо показывают суть основанных на них улучшенных методов. Кроме того, в некоторых случаях (как правило, при небольшой длине массива и/или особом исходном расположении элементов массива) некоторые из прямых методов могут даже превзойти улучшенные методы.

5.3. Сортировка вставкой

Принцип метода:

Массив разделяется на две части: отсортированную и неотсортированную. Элементы из неотсортированной части поочередно выбираются и вставляются в отсортированную часть так, чтобы не нарушать в ней упорядоченность элементов. В начале работы алгоритма в качестве отсортированной части массива принимают только один первый элемент, а в качестве неотсортированной части — все остальные элементы.

Таким образом, алгоритм будет состоять из (n - 1)-го прохода (n — размерность массива), каждый из которых будет включать четыре действия:

— взятие очередного i-го неотсортированного элемента и сохранение его в дополнительной переменной;

— поиск позиции j в отсортированной части массива, в которой присутствие взятого элемента не нарушит упорядоченности элементов;

— сдвиг элементов массива от (i - 1)-го до (j - 1)-го вправо, чтобы освободить найденную позицию вставки;

— вставка взятого элемента в найденную j-ю позицию.

Программа, реализующая рассмотренный алгоритм, будет иметь следующий вид:

```
ClrList↵
"Введите длину массива " : ? →N ↵
For 1→I To N ↵
"Введите" : I◢
"-й элемент массива" ↵
?→List 1 [ I ] ↵
Next ↵
List 1◢
For 2→I To N ↵
List 1 [ I ] →B↵
1→J ↵
While B> List 1 [ J ] ↵
J+1→J ↵
WhileEnd↵
For I-1→K To J Step -1 ↵
List 1 [ K ] → List 1 [ K+1 ] ↵
Next ↵
B →List 1 [ J ] ↵
Next ↵
List 1◢
```

5.4. Сортировка выбором

Принцип метода:

Находим (выбираем) в массиве элемент с минимальным значением на интервале от 1-го элемента до *n*-го (последнего) элемента и меняем его местами с первым элементом. На втором шаге находим элемент с минимальном значением на интервале от 2-го до *n*-го элемента и меняем его местами со вторым элементом.

И так далее для всех элементов до (*n* - 1)-го.

Программа, реализующая метод выбора, будет иметь следующий вид:

```
ClrList↵
"Введите длину массива " : ? →N ↵
For 1→I To N ↵
"Введите" : I◢
"-й элемент массива" ↵
?→List 1 [ I ] ↵
Next ↵
List 1◢
For 1→S To N-1 ↵
List 1 [S ] →M ↵
S→T ↵
For S +1→I To N ↵
If List 1 [ I ]<M ↵
Then List 1 [ I ] →M↵
I→T ↵
IfEnd ↵
Next ↵
List 1 [ S ] → List 1 [ T ] ↵
M→ List 1 [S ] ↵
Next ↵
List 1◢
```

5.5. Сортировка обменом («пузырьковая» сортировка)

Принцип метода:

Слева направо поочередно сравниваются два соседних элемента, и если их взаиморасположение не соответствует заданному условию упорядоченности, то они меняются местами. Далее берутся два следующих соседних элемента и так далее до конца массива.

После одного такого прохода на последней *n*-ой позиции массива будет стоять максимальный элемент («всплыл» первый «пузырек»). Поскольку максимальный элемент уже стоит на своей последней позиции, то второй проход обменов выполняется до (*n* - 1)-го элемента. И так далее. Всего требуется (*n* - 1) проход.

Программа, реализующая метод обмена («пузырька»), будет иметь следующий вид:

```
ClrList↵
"Введите длину массива " : ? →N ↵
For 1→I To N ↵
"Введите" : I◢
"-й элемент массива" ↵
?→List 1 [ I ] ↵
Next ↵
List 1◢
For N→K To 2 Step -1 ↵
For 1→I To K-1 ↵
If List 1 [ I ]< List 1 [ I+1 ] ↵
Then List 1 [ I ] →B↵
List 1 [ I+1 ] → List 1 [ I ] ↵
B→ List 1 [ I+1 ] ↵
IfEnd ↵
Next ↵
Next ↵
List 1◢
```

ЯПК имеет встроенную функцию сортировки (Sort), которая существенно облегчает решение подобных задач.

Пример 22.

Заполнить одномерный массив из *n* элементов целых чисел. Упорядочить его:

а) по возрастанию;

б) по убыванию

```
ClrList↵
"Введите длину массива " : ? →N ↵
For 1→I To N ↵
"Введите элемент массива " : I◢
?→List 1 [ I ] ↵
Next ↵
SortA(List 1) ↵ /по возрастанию/
List 1◢
SortD(List 2) ↵ /по убыванию/
List 1◢
```

Задачи

87. Заполнить массив из *n* элементов целых чисел, упорядочить его:

а) по возрастанию;

б) по убыванию.

88. Заполнить одномерный массив из n элементов чисел. Упорядочить первую половину элементов по возрастанию, вторую — по убыванию.

89. Заполнить одномерный массив из n элементов чисел. Упорядочить первую половину элементов по убыванию, вторую — по возрастанию.

90. Заполнить одномерный массив из n элементов чисел. Необходимо переставить элементы массива так, чтобы в начале шла группа элементов, больших того, который в исходном массиве располагался на первом месте, затем сам этот элемент, потом — группа элементов, меньших или равных ему.

91. Заполнить одномерный массив из n элементов чисел. Найти наибольший элемент, встречающийся в массиве после выбрасывания максимального элемента.

92. Ввести два одномерных массива с различным количеством элементов и натуральное число k. Объединить их в один массив, включив второй массив между k-м и $(k+1)$-м элементами первого, не используя дополнительный массив.

93. В массиве A каждый элемент равен 0, 1 или 2. Переставить элементы массива так, чтобы сначала располагались все единицы, затем все двойки и, наконец, все нули (дополнительного массива не заводить).

94. Заполнить одномерный массив из n элементов чисел. Упорядочить элементы массива, стоящие на нечетных местах, в возрастающем порядке, а на четных — в убывающем.

95. Из двух упорядоченных одномерных массивов (длины K и N) сформируйте одномерный массив размером K+N, упорядоченный так же, как исходные массивы.

96. Ввести целочисленный массив. Сформировать второй массив всех таких различных значений, которые в первом массиве встречаются по два и более раза.

6. ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ ЯПК

В современных компьютерах имеются два основных режима работы монитора: текстовый и графический. Характерной особенностью калькуляторов является то, что изначально устанавливается графический режим работы дисплея.

ЯПК не предназначен для создания произвольных картинок: дисплей калькулятора монохромный и небольшого размера (127×63 точки).

Прежде чем начинать графические построения, целесообразно ввести операторы очистки (ClrText, ClrList, ClrGraph), чтобы они не накладывались на ранее вручную выполненные построения.

Отличительной особенностью графического режима ЯПК является построение относительно начала координат и автоматическое построение осей координат. ЯПК позволяет изменять масштаб графического изображения. При увеличении интервала, на котором будет рассматриваться изображение, точность построений будет уменьшаться, при уменьшении интервала — возрастать. Если требуется убрать оси координат, то можно, например, сдвинуть их так, чтобы левый нижний угол имел координаты (0, 0).

Изменять масштаб изображения можно как в самой программе, так и в ходе выполнения программы. Чтобы масштаб был постоянен, в программе вводится:

ViewWindow <Xmin, Xmax, Xscale, Ymin, Ymax, Yscale>

где: Xmin — минимальное значение по оси OX;
Xmax — максимальное значение по оси OX;
Xscale — расстояние единичного отрезка по оси OX;
Ymin — минимальное значение по оси OY;
Ymax — максимальное значение по оси OY;
Yscale — расстояние единичного отрезка по оси OY.

Изменение масштаба в ходе выполнения программы может осуществляется следующим образом:

```
ClrGraph↵
"OX=" : ? →A ↵
"OY=" : ? →B ↵
ViewWindow -A, A, 1, -B, B, 1 ↵
... ... ...
Либо
"Xmin=" : ? →A ↵
"Xmax=" : ? →B ↵
"ед.OX=" : ? →C ↵
```

"Ymin" : ? →D ↵
"Ymax" : ? →E ↵
"ед.OY=" : ? →F ↵
ViewWindow A, B, C, D, E, F ↵
... ...

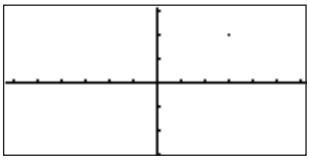
По умолчанию ЯПК вводит стандартные установки окна: Xmin=-6,3, Xmax=6,3, Xscale=1, Ymin=-3,1, Ymax=3,1, Yscale=1.

Процедура PlotOn выводит точку по указанным координатам: PlotOn X, Y. Здесь X, Y — координаты точки относительно начала координат.

Пример 23.

Программа, строящая точку с координатами (3, 2).

PlotOn 3, 2



В ЯПК имеются две процедуры построения линий.

1. Процедура F-Line строит линию с указанными координатами начала и конца: F-Line X1, Y1, X2, Y2. Здесь X1, Y1 — координаты начала, X2, Y2 — координаты конца линии относительно начала координат.

2. Процедура Line строит линию по указанным точкам начала и конца:

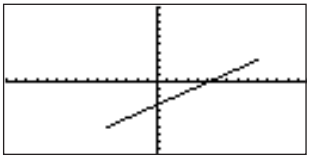
Plot X1, Y1↵
Plot X2, Y2↵
Line

Здесь Plot X1, Y1 — начальная точка, Plot X2, Y2 — конечная точка.

Пример 24.

Программа, строящая линию с координатами (-5; -6) и (10; 3).

ViewWindow -15, 15, 1, -10, 10, 1 ↵
F-Line -5, -6, 10, 3



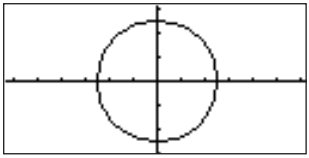
Процедура Circle строит окружность: Circle X, Y, R. Здесь X, Y — координаты центра относительно начала координат; R — радиус в

пикселях. Изменяя масштаб изображения на экране, можно получить эллипс.

Пример 25.

Программа, строящая окружность с центром в начале координат и радиусом 2,5.

```
ClrGraph↵
Circle 0, 0, 2.5
```



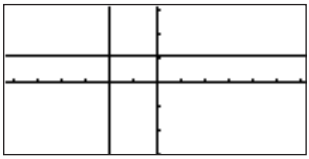
Процедура Vertical вычерчивает вертикальную линию: Vertical X.

Процедура Horizontal вычерчивает горизонтальную линию: Horizontal Y.

Пример 26.

Программа, строящая линии, параллельные осям координат.

```
ClrGraph ↵
Vertical -2 ↵
Horizontal 1.1
```

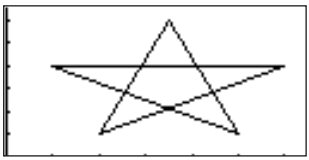


В графическом режиме можно создавать иллюзию простейших образов.

Пример 27.

Программа, изображающая в центре экрана пятиконечную звезду.

```
ClrGraph ↵
ViewWindow 0, 6.5, 1, 0, 6.5, 1 ↵
F-Line 3.5, 6, 2, 1↵
F-Line 3.5, 6, 5, 1↵
F-Line 2, 1, 6, 4↵
F-Line 5, 1, 1, 4↵
F-Line 1, 4, 6, 4↵
```



Аналогичным образом можно строить любые геометрические фигуры.

Можно создавать иллюзию движения.

Пример 28.

Движение точки по экрану.

```
ClrGraph↵
For 1→I To 5 Step 0.1 ↵
ClrGraph↵
PlotOn -2, -2+I↵
Next
```

Пример 29.

Движение точки, которая оставляет за собой след.

```
ClrGraph↵
For 1→I To 5 Step 0.1 ↵
PlotOn -2, -2+I↵
Next
```

Пример 30.

Движущиеся друг за другом прямоугольник и окружность.

```
ClrGraph↵
For 1→I To 4 Step 0.1 ↵
Circle -5+I, 0, 2↵
F-Line -2+I, -2, -2+I, 2↵
F-Line 1+I, -2, 1+I, 2↵
F-Line -2+I, -2, 1+I, -2↵
F-Line -2+I, 2, 1+I, 2↵
ClrGraph↵
Next
```

Процедура Text позволяет вставить текст в графическом режиме: Text X, Y "текст", где X, Y — начальные координаты ввода текста.

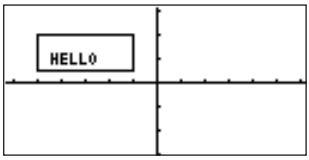
Текст может быть задан лишь латинскими буквами. Номер строки задается в диапазоне от 1 до 63, а номер столбца — в диапазоне от 1 до 127 и не зависит от осей координат.

Пример 31.

Программа, вставляющая слово HELLO в прямоугольник.

```
ClrGraph↵
Plot -5, 0.5↵
Plot -5, 2↵
Line↵
Plot -5, 2↵
Plot -1, 2↵
Line↵
Plot -5, 0.5↵
```

```
Plot -1, 0.5↵
Line↵
Plot -1, 0.5↵
Plot -1, 2↵
Line↵
Text 20, 20, "HELLO"
```



Целесообразно применять данную процедуру для подписи графиков функций.

ЯПК обладает широкими возможностями в области построения и исследования функций. Рассмотрим подробнее построение графиков функций.

Графики можно строить в прямоугольных и полярных координатах. В прямоугольных координатах функция записывается в виде: $Y = f(X)$, где значение координаты по оси Y вычисляется по заданной формуле для каждого значения координаты X. В полярных координатах функция записывается в виде: $r = f(\theta)$, где r «радиус» — расстояние от начала координат, θ — угол (определяется так же, как в тригонометрии).

Процедура Graph Y= позволяет построить график функции в декартовых координатах.

Процедура Graph r= позволяет построить график функции в полярных координатах.

Для построения графиков нескольких функций целесообразно воспользоваться следующей процедурой:

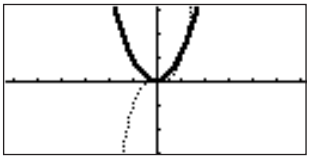
```
Y=Type ↵
" f1(X)" → Y1↵
" f2(X)" → Y2↵
... ... ...
" fn(X)" → Yn↵
DrawGraph
```

Использование данной процедуры позволяет изменять стиль линии графика: NormalG n, ThickG n, BrokenThickG n, DotG n, где n — номер соответствующей функции. ЯПК позволяет выводить на экран формулу функции, график которой строится в настоящий момент.

Пример 32.

Программа, строящая сплошной толстой линией график функции $y = x^2$ и точками — график функции $y = x^3$.

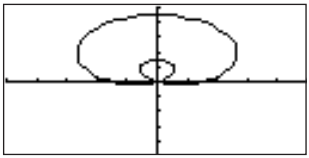
```
ClrGraph↵
Y=Type ↵
" X²" → Y1↵
" X^3" → Y2↵
ThickG 1↵
DotG 2 ↵
DrawGraph
```



Пример 33.

Программа, строящая график функции $r = -1{,}5 + 3\sin\theta$ в полярной системе координат.

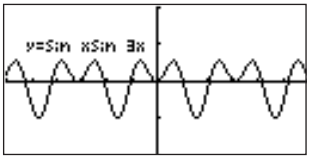
```
ClrGraph↵
"OX=" : ? →A ↵
"OY=" : ? →B ↵
ViewWindow -A, A, 1, -B, B, 1 ↵
Graph r=-1.5+3sinθ↵
```



Пример 34.

Программа, строящая подписанный график функции $y = \sin x \cdot \sin 3x$.

```
ClrGraph↵
ViewWindow -6, 6, 1, -2, 3,
1 ↵
Graph Y=sinXsin3X↵
Text 15, 10, "y=sinxsin3x"
```



Задачи

97. Построить прямоугольник и вписать в него свое имя.

98. Составить программу, выводящую на дисплей свое имя с помощью отрезков, прямоугольников и окружностей.

99. Построить в декартовых координатах график функции $y = x^5 - 3x^4 + 2x^3 - 7x^2 + 4x - 8$.

100. Построить в полярных координатах график функции $r = 2\sin 3\theta$.

101. Построить график функции, заданной следующими уравнениями: $x = 7\cos T - 2\cos 3{,}5T$; $y = 7\sin T - 2\sin 3{,}5T$.

102. Построить правильный многоугольник с заданным числом сторон.

103. Построить параллелограмм с проведенными в нем диагоналями.

104. Построить произвольную фигуру, состоящую из графических примитивов.

105. Составить программу, выводящую на дисплей калькулятора визитную карточку учебного заведения.

106. Составить программу, выводящую на дисплей изображение ведерка.

107. Написать программу, создающую иллюзию движущегося автомобиля.

108. Написать программу, создающую иллюзию движущейся окружности с проведенным в ней диаметром.

109. Написать программу, имитирующую изображение лица.

7. СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ

ЯПК позволяет разрабатывать программы с использованием идеологии структурного языка программирования. В этом состоит одно из его достоинств с точки зрения обучения программированию.

Напомним, что основная идея структурного программирования состоит в том, что одну большую программу можно разбить на небольшие относительно самостоятельные части, которые можно разрабатывать и отлаживать отдельно или по частям. Затем большую программу можно собрать как бы по кирпичикам. Одно из преимуществ структурного программирования заключается в том, что подпрограмма описывается один раз, а вызывается столько, сколько потребуется. В ЯПК такие части будем называть подпрограммами.

Подпрограмма представляет собой относительно самостоятельную программу с определенным именем. Упоминание этого имени следующим образом в тексте программы называется вызовом подпрограммы:

```
...
Prog "имя подпрограммы"
...
```

Пример 35.

Программа, имитирующая попарное мигание двух кубиков в разных частях экрана.

```
Подпрограмма "KUB1"
        ViewWindow -20, 20, 1, -15, 15, 1 ↵
        F-Line 1, 1, 1, 2.5 ↵
        F-Line 1, 2.5, 3, 2.5 ↵
        F-Line 3, 1, 1, 1 ↵
        F-Line 3, 2.5, 3, 1 ↵
        ClrGraph
Подпрограмма "KUB2"
        ViewWindow -20, 20, 1, -15, 15, 1 ↵
        F-Line 10, 8, 10, 9.5 ↵
        F-Line 10, 9.5, 13, 9.5 ↵
        F-Line 13, 9.5, 13, 8 ↵
        F-Line 13, 8, 10, 8 ↵
        ClrGraph
Программа MIG
        Lbl 1↵
        Prog "KUB1"↵
```

```
    Prog "KUB2"↵
    Goto 1
```

Задачи существенно упрощаются, если использовать процедуры с параметрами. Усложним предыдущую программу, добавив условие: мигающие кубики должны увеличиваться в размере.

Пример 36.

```
Подпрограмма "KUBIK1"
    ViewWindow -20, 20, 1, -15, 15, 1 ↵
    F-Line 1-I, 1-I, 1-I, 2.5+I ↵
    F-Line 1-I, 2.5+I, 3+I, 2.5+I ↵
    F-Line 3+I, 1-I, 1-I, 1-I ↵
    F-Line 3+I, 2.5+I, 3+I, 1-I ↵
    ClrGraph
Подпрограмма "KUBIK2"
    ViewWindow -20, 20, 1, -15, 15, 1 ↵
    F-Line 10-I, 8-I, 10-I, 9.5+I ↵
    F-Line 10-I, 9.5+I, 13+I, 9.5+I ↵
    F-Line 13+I, 9.5+I, 13+I, 8-I ↵
    F-Line 13+I, 8-I, 10-I, 8-I ↵
    ClrGraph
Программа YVEL
    For 1→I To 5 Step 0.2 ↵
    Prog " KUBIK 1"↵
    Prog " KUBIK 2"↵
    Next
```

Рассмотрим пример возведения любого числа в любую степень.

Пример 37.

```
Программа PARAM1
    "Введите число":?→X ↵
    "Введите показатель степени":?→Y ↵
    X→A ↵
    Y→B ↵
    Prog "STEPEN" ↵
    "Степень =":S◢
Подпрограмма "STEPEN"
    If A>0↵
    Then e^(B×ln A) →S↵
    Else If A<0↵
    Then e^(B×ln (Abs A)) → S↵
    Else If 0→B
    Then 1→ S↵
    Else 0→ S↵
    IfEnd↵
```

IfEnd↵
IfEnd

В программе вводится пара чисел X и Y и выводится на экран результат возведения числа X в степень Y.

Параметры X, Y в момент обращения к подпрограмме "STEPEN" — это фактические параметры. Они подставляются вместо формальных параметров A и B, заданных в основной программе, и затем над ними осуществляются нужные действия. Полученный результат присваивается идентификатору подпрограммы — именно он и будет возвращен как значение при выходе из нее.

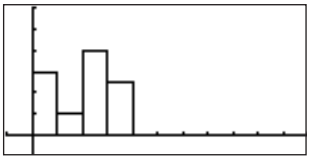
Характерной особенностью ЯПК является то, что все используемые параметры являются глобальными по отношению ко всем подпрограммам. Все программы являются глобальными, проверяются ЯПК лишь на наличие, место их описания не принципиально. В ЯПК возможна рекурсия, т.е. программа, может вызывать саму себя.

Пример 38.

Программа, строящая столбчатую диаграмму по заданным значениям.

Подпрограмма "DANN"
"Введите": I◢
"-й элемент данных": ? →List 1 [I]
Подпрограмма "STOLB"
F-Line 1×I, 0, 1×I, 0+ List 1 [I] ↵
F-Line 1×(I-1), 0, 1×(I-1), 0+ List 1 [I] ↵
F-Line 1×(I-1), 0+ List 1 [I], 1×I, 0+ List 1 [I]
Программа DIAGRAM
ViewWindow -1, 11, 1, -1, 6, 1 ↵
"Количество данных":?→N ↵
For 1→ I To N↵
Prog "DANN" ↵
Prog "STOLB" ↵
Next

Для данных: 3; 1; 4; 2,5 имеем следующую диаграмму:



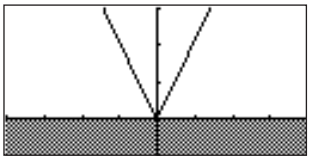
Пример 39.

Программа, демонстрирующая закон отражения света. Вводится значение угла падения, затем выполняется построение хода падающего луча, затем отраженного и перпендикуляра к отражающей по-

верхности в точке падения луча, далее выводится значение угла отражения.

```
ViewWindow -4, 4, 1, -1, 3, 1 ↵
Graph Y<0◢
Lbl 1:ClrText↵
"A°=":?→A ↵
A<0 Or A>90 ⇒ Goto 1 ↵
Graph Y=Abs(X÷tanA)◢
ClrText↵
Graph X=0◢
Cls↵
ClrText↵
Locate 1,1, "A=_ _°"↵
Locate 3,1, A↵
Locate 17,1, "B=_ _°"↵
Locate 19,1, A
```

Для угла падения 35° имеем:



Использование структурного программирования позволяет создавать более сложные, динамические программные проекты.

Пример 40.

Программа, которая создает на экране окно и имитирует движение в нем кубика под углом к поверхности. При столкновении с гранью окна кубик отскакивает от нее по закону отражения.

Подпрограмма "KUBIK"

```
F-Line X, Y, X, Y+1 ↵
F-Line X, Y, X+1, Y ↵
F-Line X+1, Y+1, X+1, Y ↵
F-Line X+1, Y+1, X, Y ↵
```

Подпрограмма "OKNO"

```
F-Line -6, 3, 6, 3 ↵
F-Line 6, 3, 6, -3 ↵
F-Line 6, -3, -6, -3 ↵
F-Line -6, -3, -6, 3
```

Подпрограмма "DVIG"

```
Prog "OKNO" ↵
X+A→X ↵
Y+B→Y ↵
```

```
        Prog "KUBIK" ↵
        ClrGraph
Программа DVIGENIE
        ClrGraph ↵
        -6→X ↵
        -6→Y ↵
        0.2→A ↵
        0.2→B ↵
        Lbl 1 ↵
        Prog "DVIG" ↵
        If (X≤-6) And (A=-0.2) And (B=0.2) ↵
        Then 0.2→A ↵
        IfEnd ↵
        If (X≤-6) And (A=-0.2) And (B=-0.2) ↵
        Then 0.2→A ↵
        IfEnd ↵
        If (Y≥2) And (A=0.2) And (B=0.2) ↵
        Then -0.2→B
        IfEnd ↵
        If (Y≥2) And (A=-0.2) And (B=0.2) ↵
        Then -0.2→B
        IfEnd ↵
        If (X≥5) And (A=0.2) And (B=0.2) ↵
        Then -0.2→A
        IfEnd ↵
        If (X≥5) And (A=0.2) And (B=-0.2) ↵
        Then -0.2→A
        IfEnd ↵
        If (Y≤-3) And (A=0.2) And (B=0.2) ↵
        Then 0.2→B ↵
        IfEnd ↵
        If (Y≤-3) And (A=-0.2) And (B=-0.2) ↵
        Then 0.2→B ↵
        IfEnd ↵
        Goto 1
```

Задачи

110. Создать программу, имитирующую на дисплее калькулятора стакан с чаем, в который падает кусочек сахара. Пока сахар тонет, он тает. Около нижней грани он исчезает.

111. Создать программу, имитирующую движение в окне окружности; при столкновении с гранью окна она отскакивает по закону отражения.

112. Создать программу, имитирующую попарное мигание трех окружностей в разных частях экрана.

113. Создать программу, имитирующую полет летающей тарелки.

114. Создать программу, имитирующую движение нескольких кубиков по дисплею калькулятора.

115. Создать программу, имитирующую закон преломления света.

116. Число медалистов школы в 2003 г. составило 15 выпускников, в 2004 г. — 17 выпускников, в 2005 г. — 20 выпускников, в 2007 г. — 14 выпускников. Построить по этим данным столбчатую диаграмму.

117. Построить пузырьковую диаграмму по исходным данным.

118. Определить дальность полета тела, брошенного под углом к горизонту. Построить график зависимости движения тела от времени.

119. Создать программу, имитирующую броуновское движение для двух частиц с поочередным построением траектории их движения.

120. Решить уравнение $5x^2 - 3x - 2 = 0$. Построить графическую интерпретацию решения уравнения.

121. Решить уравнение $x^2 - 4\frac{1}{2}x + 4\frac{1}{2} = 0$. Построить графическую интерпретацию решения уравнения.

122. Решить уравнение $\frac{x^2}{3} - 0{,}5 = 0$. Построить графическую интерпретацию решения уравнения.

123. С заданной точностью решить уравнение $x^2 = \sin x$. Построить графическую интерпретацию решения уравнения.

124. Решить систему уравнений: $\begin{cases} \frac{5}{2}x + \frac{1}{5}y = -4, \\ \frac{1}{3}x + \frac{1}{6}y = \frac{1}{6}. \end{cases}$

Построить графическую интерпретацию решения.

125. Решить систему уравнений: $\begin{cases} x + 2y = 3, \\ 4x + 5y = 6. \end{cases}$

Построить графическую интерпретацию решения.

Приложение

Оператор, функция	Последовательность нажатия клавиш для доступа к данному оператору					
If	SHIFT	PRGM	F1	F1		
Then	SHIFT	PRGM	F1	F2		
Else	SHIFT	PRGM	F1	F3		
IfEnd	SHIFT	PRGM	F1	F4		
For	SHIFT	PRGM	F1	F6	F1	
To	SHIFT	PRGM	F1	F6	F2	
Step	SHIFT	PRGM	F1	F6	F3	
Next	SHIFT	PRGM	F1	F6	F4	
While	SHIFT	PRGM	F1	F6	F6	F1
WhileEnd	SHIFT	PRGM	F1	F6	F6	F2
Do	SHIFT	PRGM	F1	F6	F6	F3
LpWhile	SHIFT	PRGM	F1	F6	F6	F4
Prog	SHIFT	PRGM	F2	F1		
Return	SHIFT	PRGM	F2	F2		
Break	SHIFT	PRGM	F2	F3		
Stop	SHIFT	PRGM	F2	F4		
Lbl	SHIFT	PRGM	F3	F1		
Goto	SHIFT	PRGM	F3	F2		
⇒	SHIFT	PRGM	F3	F3		
Isz	SHIFT	PRGM	F3	F4		
Dsz	SHIFT	PRGM	F3	F5		
?	SHIFT	PRGM	F4			
◢	SHIFT	PRGM	F5			
:	SHIFT	PRGM	F6	F5		
ClrText	SHIFT	PRGM	F6	F1	F1	
ClrGraph	SHIFT	PRGM	F6	F1	F2	
ClrList	SHIFT	PRGM	F6	F1	F3	
ClrMat	SHIFT	PRGM	F6	F1	F4	
=	SHIFT	PRGM	F6	F3	F1	
≠	SHIFT	PRGM	F6	F3	F2	
>	SHIFT	PRGM	F6	F3	F3	

Оператор, функция	Последовательность нажатия клавиш для доступа к данному оператору					
<	SHIFT	PRGM	F6	F3	F4	
≥	SHIFT	PRGM	F6	F3	F5	
≤	SHIFT	PRGM	F6	F3	F6	
Locate	SHIFT	PRGM	F6	F4	F1	
x!	OPTN	F6	F3	F1		
Ran#	OPTN	F6	F3	F4		
Abs	OPTN	F6	F4	F1		
Int	OPTN	F6	F4	F2		
Frac	OPTN	F6	F4	F3		
Rnd	OPTN	F6	F4	F4		
Intg	OPTN	F6	F4	F5		
And	OPTN	F6	F6	F4	F1	
Or	OPTN	F6	F6	F4	F2	
Not	OPTN	F6	F6	F4	F3	
List	OPTN	F1	F1			
Sort A	F4	F3	F1			
Sort D	F4	F3	F2			
Min	OPTN	F1	F6	F1		
Max	OPTN	F1	F6	F2		
Draw Graph	SHIFT	PRGM	F6	F2	F2	
View Window	SHIFT	F3	F1			
Y=Type	F4	F4	F3	F1		
r=Type	F4	F4	F3	F2		
NormalG	F4	F4	F4	F1		
ThickG	F4	F4	F4	F2		
DotG	F4	F4	F4	F4		
StoGMEN	F4	F4	F5	F1		
RclGMEN	F4	F4	F5	F2		
Cls	SHIFT	F4	F1			
Graph Y=	SHIFT	F4	F5	F1		
Graph r=	SHIFT	F4	F5	F2		
Graph X=	SHIFT	F4	F5	F4		
Graph Y>	SHIFT	F4	F5	F6	F1	

Оператор, функция	Последовательность нажатия клавиш для доступа к данному оператору					
Graph Y<	SHIFT	F4	F5	F6	F2	
Graph Y≥	SHIFT	F4	F5	F6	F3	
Graph Y≤	SHIFT	F4	F5	F6	F4	
Plot	SHIFT	F4	F6	F1	F1	
PlotOn	SHIFT	F4	F6	F1	F2	
Line	SHIFT	F4	F6	F2	F1	
Circle	SHIFT	F4	F6	F3		
Vertical	SHIFT	F4	F6	F4		
Horizontal	SHIFT	F4	F6	F5		
Text	SHIFT	F4	F6	F6	F2	
PxlOn	SHIFT	F4	F6	F6	F3	F1
PxiOff	SHIFT	F4	F6	F6	F3	F2

ОГЛАВЛЕНИЕ

Учебное издание

Востроknутов Игорь Евгеньевич,
Помелова Мария Сергеевна

**Учимся программировать
на графических калькуляторах
CASIO FX-9860G**

**Учебное пособие для общеобразовательных
учебных заведений**

Ответственный редактор *А.Б.Игрунов*
Художественное оформление: *А.В.Аксенов*
Компьютерная верстка: *Л.А.Вишнякова*

Подписано в печать 14.03.2008. Формат 60×90/16.
Бумага офсетная. Печать офсетная. Гарнитура «Таймс». Усл. печ. л. 3,75.
Тираж 1000 экз. Заказ № 225

Издательско-полиграфическая фирма ООО «Принтберри».
Москва, пр-д Комсомольской пл., 12.
Тел./факс: (495) 545-4764. e-mail: info@printberry.ru