МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ БЕЛОРУССКОЙ ССР
РЕСПУБЛИКАНСКИЙ УЧЕБНО-МЕТОДИЧЕСКИЙ КАБИНЕТ
ГОМЕЛЬСКИЙ ОБЛАСТНОЙ ИНСТИТУТ УСОВЕРШЕНСТВОВАНИЯ УЧИТЕЛЕЙ
ГОМЕЛЬСКИЙ ОБЛАСТНОЙ СОВЕТ ПЕДАГОГИЧЕСКОГО ОБЩЕСТВА

ПЭВМ «АГАТ» В УПК И ШКОЛЕ

Гомель · 1988

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ БЕЛОРУССКОЙ ССР
РЕСПУБЛИКАНСКИЙ УЧЕБНО-МЕТОДИЧЕСКИЙ КАБИНЕТ
ГОМЕЛЬСКИЙ ОБЛАСТНОЙ ИНСТИТУТ УСОВЕРШЕНСТВОВАНИЯ УЧИТЕЛЕЙ
ГОМЕЛЬСКИЙ ОБЛАСТНОЙ СОВЕТ ПЕДАГОГИЧЕСКОГО ОБЩЕСТВА БССР

П Э В М "А Г А Т" В У П К И Ш К О Л Е

(Учебное пособие для учащихся)

Под редакцией В.М.Лайхтмана

Гомель · 1988

УДК 681.3 (07)

ПЭВМ "АГАТ" в УПК и школе: (Учебное пособие для учащихся)/Составители: О.Г.Свиридова, И.В.Ульянова, Л.Б.Каган, Г.А.Савидова, Р.П.Гороховская, Ж.Л.Песина, В.Д.Вершинина.
Под ред. В.М. Л а й х т м а н а. - Гомель, 1988. - 64 с.

Учебное пособие разработано на основе двухлетнего опыта преподавания информатики с использованием ПЭВМ "АГАТ" в УПК.

Приведены начальные сведения об устройстве и функциональной роли основных блоков ПЭВМ "АГАТ", о программировании на языке Бейсик версии АГАТ. Показаны вычислительные и графические возможности компьютера, способы представления и обработки текстовых данных.

Пособие может быть использовано при изучении раздела "Язык программирования Бейсик" курса "Основы информатики и вычислительной техники" учащимися средних общеобразовательных школ.

Ответственный за выпуск: А.М.Нестерчук, заслуженный учитель БССР

Р е ц е н з е н т ы: канд. физ.-мат. наук доц. М.И.Жадан (ГГУ); А.А.Лихтарович, методист отдела информатики и вычислительной техники МП БССР

Одобрено Республиканским координационным Советом Министерства просвещения Белорусской ССР

I. АРХИТЕКТУРА ПЭВМ "АГАТ". ФУНКЦИОНАЛЬНАЯ РОЛЬ ОСНОВНЫХ УСТРОЙСТВ

Персональная ЭВМ "АГАТ" представляет собой сложное устройство, которое состоит из электронных и электромеханических частей. Элементной базой ПЭВМ являются микросхемы и большие интегральные микросхемы (БИС). Персональная микро-ЭВМ "АГАТ" используется в учебных целях и ориентирована на пользователей, не имеющих профессиональных навыков работы с ЭВМ.

ПЭВМ состоит из отдельных узлов, собранных по блок-схеме, изображенной на рис.I. Конструктивно ПЭВМ состоит из следующих блоков:

I. Системного блока.
2. Видеоконтрольного устройства (ВКУ) - дисплея.
3. Мозаично-печатающего устройства (МПУ) - принтера.
4. Блока клавиатуры.

I.I. Системный блок

Системный блок является основным блоком, с помощью которого объединяются в единую систему все остальные составные части ПЭВМ.

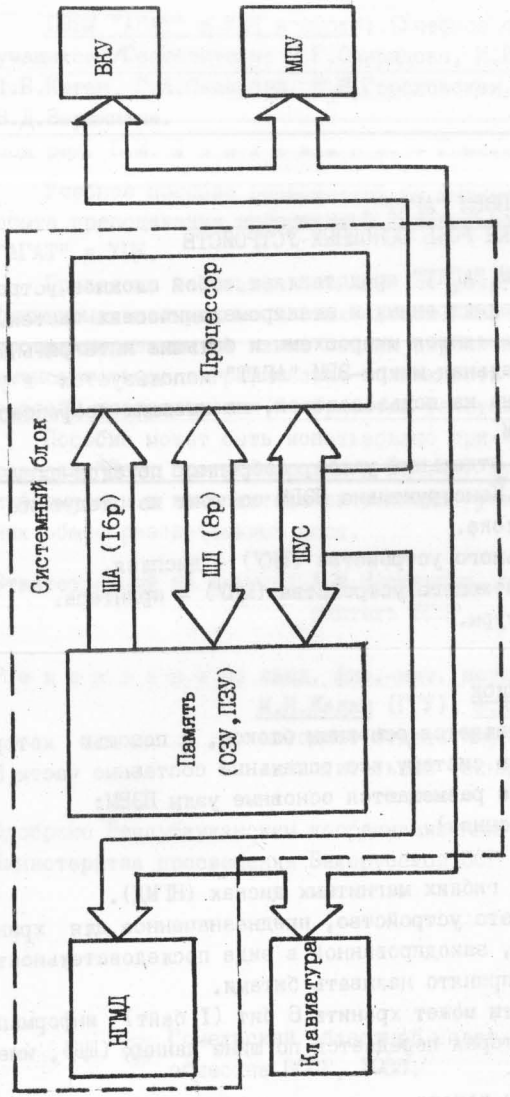
В системном блоке размещаются основные узлы ПЭВМ:

I. Память (внутренняя).
2. Процессор.
3. Накопитель на гибких магнитных дисках (НГМД).

I.I.I. Память - это устройство, предназначенное для хранения различной информации, закодированной в виде последовательностей цифр 0 и I, которые принято называть битами.

Одна ячейка памяти может хранить 8 бит (I байт) информации (машинное слово), которая передается по шине данных (ШД), имеющей 8 линий (проводов).

Объем памяти ПЭВМ "АГАТ" составляет 64 Кбайт (т.е. память содержит 64 x I024 = 65 536 ячеек). Каждая ячейка памяти имеет свой адрес. Адресация ячеек начинается с 0, а так как всего ячеек памяти



Обозначения:

ША - шина адреса.
ШД - шина данных.
ШУС - шина управляющих сигналов.
ША, ШД и ШУС объединяются в магистраль.

Рис.I. Блок-схема ПЭВМ "АГАТ"

65 536, максимальным является адрес 65 535. Для того чтобы обращаться к ячейкам по адресам, их необходимо хранить в памяти ПЭВМ и передавать между различными устройствами. Как и любая информация, адреса кодируются последовательностями нулей и единиц. Так как ячеек 65 536, то и различных комбинаций, кодирующих адреса ячеек памяти, должно быть 65 536. Получить такое количество комбинаций адресов можно с помощью I6 бит (2^{I6} = 65 536).

Адреса передаются между различными устройствами ПЭВМ с помощью I6 линий (проводов), которые объединяются в шину адреса (ША).

Память подразделяется на оперативную (ОЗУ - оперативное запоминающее устройство) и постоянную (ПЗУ - постоянное запоминающее устройство).

ОЗУ используется для временного хранения пользовательских программ, исходных данных, промежуточных и конечных результатов.

ПЗУ используется для длительного хранения управляющих программ.

I.I.2. Процессор - это сложное электронное устройство, которое выполняет следующие функции:

I. Обеспечивает автоматическое выполнение программы.

2. Осуществляет обработку информации (выполнение арифметических, логических и других операций).

3. Руководит работой всех устройств ПЭВМ с помощью управляющих сигналов. Управляющие сигналы передаются ко всем устройствам ПЭВМ по линиям (проводам), объединяющимся в шину управляющих сигналов (ШУС).

I.I.3. Накопитель на гибких магнитных дисках (НГМД). Он представляет собой электронно-механическое устройство. Накопитель является внешним устройством ПЭВМ. С его помощью можно осуществить ввод и вывод информации с внешнего носителя - гибкого магнитного диска (ГМД).

НГМД размещается в системном блоке и расположен в правом углу в виде темного окошка с прорезью, куда вставляется гибкий магнитный диск.

При выключении машины вся информация, хранящаяся в памяти, теряется. Для того чтобы каждый сеанс работы пользователя с ПЭВМ не приходилось начинать с ввода заново необходимых программ и исходных данных, используется ГМД. Он позволяет сохранить информацию между сеансами работы на ЭВМ длительное время.

НГМД - это устройство, которое позволяет осуществлять запись информации из памяти машины на ГМД и чтение информации с ГМД во внутреннюю память машины.

Гибкий магнитный диск является внешней памятью машины и представляет собой круглую пластмассовую пленку, покрытую специальным веществом, обладающим магнитными свойствами (похожим составом покрыта магнитофонная лента). ГМД помещен в квадратный плоский конверт из плотной черной бумаги, предохраняющий гибкий диск от повреждений. Центральное круглое отверстие, имеющееся на диске и конверте, предназначено для того, чтобы НГМД обеспечил вращение ГМД внутри конверта. Кроме того, в конверте есть продолговатое отверстие вдоль радиуса диска, через которое осуществляется контакт магнитной головки НГМД с поверхностью диска. Когда диск вставляется в накопитель, центральное круглое отверстие в нем надевается на вращающуюся ось. Закрытая крышка прижимает диск к оси, в результате чего диск начинает вращаться, совершая несколько оборотов в секунду. После этого через прорезь в бумажном конверте магнитная головка, приближаясь к поверхности диска, осуществляет запись новой или считывание ранее записанной на диск информации.

Запись и считывание информации происходит так же, как в магнитофонах, только записывается и воспроизводится не музыка, а цифровая информация, представленная в виде последовательностей нулей и единиц, кодирующих нужную информацию. Магнитная головка свободно перемещается по радиусу диска (через прорезь в конверте) и для того, чтобы подвести головку к нужному месту вращающегося диска, необходимо поместить её на заданное расстояние от центра и подождать, пока нужная часть диска, вращаясь, не подойдет к ней. Это происходит за доли секунды.

Благодаря возможности сравнительно быстрого доступа к любой части записанной на диске информации, НГМД является быстродействующим внешним устройством ПЭВМ.

Правила пользования ГМД:

1. Не касаться магнитного покрытия диска.
2. Не сгибать и не складывать диск.
3. Правильно вставлять диск в НГМД: этикеткой - вверх, прорезью для магнитной головки - вперед.

I.2. Видеоконтрольное устройство (дисплей)

Неотъемлемым атрибутом диалогового режима работы является наличие вывода информации из ПЭВМ на то или иное устройство отображения. В ПЭВМ "АГАТ" для этих целей служит цветное видеоконтрольное

устройство (ВКУ). Отображение информации на экране дисплея осуществляется в виде алфавитно-цифровых символов либо в графическом виде в форме рисунков, картинок, графиков.

Изображение на экране дисплея состоит из отдельных точек. Оно строится таким же образом, как и в обычном телевизоре.

I.3. Мозаично-печатающее устройство (принтер)

МПУ является внешним устройством (устройством вывода) и используется для получения твердой копии информации, хранимой в ПЭВМ, или для вывода результатов на бумагу. С его помощью можно печатать тексты, рисунки, графики. Все символы, которые может напечатать принтер, состоят из отдельных точек, расстояние между которыми очень мало. Поэтому любой символ, рассматриваемый на расстоянии, кажется единым целым. Печать символов осуществляется с помощью печатающей головки, перемещающейся относительно бумаги. Она содержит пучок иголок, из которых формируется конфигурация того или иного символа. Иголочки, ударяя по красящей ленте, оставляют на бумаге красящий след.

I.4. Блок клавиатуры

Блок клавиатуры является основным устройством, с помощью которого осуществляется общение пользователя с ПЭВМ. Одним из отличительных признаков персональной ЭВМ является то, что выполнение большей части программ происходит в диалоговом режиме работы при непосредственном активном участии человека. В этом случае блок клавиатуры используется для оперативного ввода директив и новой информации в ПЭВМ в ответ на её запросы.

В основном клавиатура ПЭВМ "АГАТ" напоминает клавиатуру пишущей машинки.

Порядок включения ПЭВМ "АГАТ"

1. Включить питание ВКУ на передней панели.
2. Вставить в НГМД системного блока системный ГМД (с дисковой операционной системой и интерпретатором языка Бейсик).
3. Включить питание на задней панели системного блока. При этом на блоке клавиатуры загорается лампочка индикации (справа или слева), на экране ВКУ появится надпись ** АГАТ **, а системный блок воспроизведет звуковой сигнал "бип". Начинает работать

накопитель на гибком магнитном диске (НГМД), о чем свидетельствует лампочка индикации НГМД. По окончании работы НГМД в нижнем левом углу экрана высвечивается символ] и курсор в виде мигающей черты.

Прежде чем начать диалог с ПЭВМ, необходимо изучить клавиатуру. Расположение клавиш на клавиатуре ПЭВМ "АГАТ" представлено на рис.2.

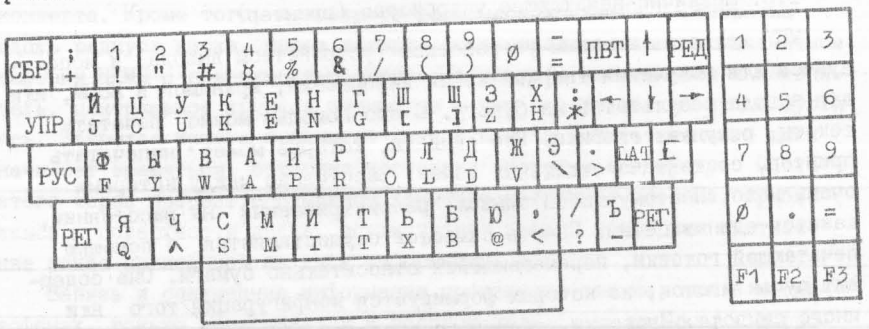


Рис.2. Клавиатура ПЭВМ "АГАТ"

Всю совокупность клавиш можно разделить на три группы:

1. Алфавитно-цифровые клавиши.
2. Функциональные клавиши.
3. Клавиши управления.

Рассмотрим подробнее назначение этих клавиш.

В набор алфавитно-цифровых клавиш входят клавиши, которые позволяют использовать в работе с ПЭВМ: 32 символа русского алфавита; 10 символов арабских цифр; 26 букв латинского алфавита; 28 символов специальных знаков (арифметических действий, скобок и т.д.).

Алфавитно-цифровые клавиши имеют на каждой клавише два символа: верхний и нижний. Верхние символы - это, в основном, цифры и русские буквы, нижние - латинские буквы и специальные знаки.

Клавиши управления. Клавиши УПРавления и СБРоса нажимаются одновременно. Нажатие этих клавиш заставляет процессор осуществить начальную установку ПЭВМ. Это означает, что нажатие клавиш УПР и СБР прекращает выполнение любой программы и выводит ПЭВМ в состояние приглашения к диалогу, высвечивая в нижнем левом углу экрана] и мигающий курсор.

Клавиша УПР защищает клавишу СБР от случайного нажатия. Нажатие одной клавиши СБР игнорируется.

Клавиша ↵ (перевод строки) обеспечивает ввод информации,

набранной на клавиатуре, в оперативную память ПЭВМ. Клавиша ↵ нажимается последней, после чего ПЭВМ воспринимает всё, что было напечатано до её нажатия и отвечает тем или иным образом.

Клавиша ПоВТорение обеспечивает повторную выдачу клавиатурой символа, нажатого вместе с клавишей ПВТ. При одновременном нажатии на клавиши ПВТ и нужного символа на экране дисплея будет осуществляться печать этого символа со скоростью 10 символов в секунду. Это повторение прекратится при отпускании либо клавиши символа, либо клавиши ПВТ.

Клавиша РЕГистр сама по себе никаких символов не генерирует. Клавиша РЕГ, нажатая одновременно с клавишей, несущей не алфавитный символ (цифру или специальный знак), позволяет напечатать специальный знак, который нанесен на соответствующую клавишу снизу.

Клавиша РЕГ, нажатая одновременно с клавишей РУССкий, позволяет набирать информацию с помощью русских символов. На клавиатуре горит красная лампочка индикации слева.

Клавиша РЕГ, нажатая одновременно с клавишей LAT (латынь), дает возможность набирать информацию с помощью латинских символов. На клавиатуре горит красная лампочка индикации справа.

Если вводятся русские символы (включен нижний регистр) и необходимо ввести одиночный латинский символ, для этого достаточно, нажав на клавишу РЕГ, ввести нужный символ. Аналогичные действия выполняются для ввода одиночного русского символа при работе с латинскими символами.

Клавиши управления курсором

↑ вверх
← влево
↓ вниз
→ вправо

Нажатие одной из этих клавиш вызывает перемещение курсора в ту сторону, в которую направлена стрелка.

Вопросы для повторения

1. На какой элементной базе выполнена ПЭВМ "АГАТ"?
2. Из каких основных блоков состоит ПЭВМ, их назначение.
3. Основные узлы системного блока.
4. Виды памяти ПЭВМ и их назначение.
5. Каковы размеры ячейки памяти, объем памяти ПЭВМ?
6. Внешние устройства ПЭВМ и их назначение.

Упражнения

I. Включить ПЭВМ "АГАТ", соблюдая указанный порядок действий, и объяснить все выполняемые машиной операции.

2. Пользуясь клавиатурой ПЭВМ, объяснить, на какие группы можно разделить совокупность всех её клавиш.

3. С помощью программы "Клавиатура ПЭВМ "АГАТ" овладеть навыками работы за клавиатурой ПЭВМ "АГАТ".

2. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ

Для работы любой ЭВМ нужен комплекс специальных программ, которые составляют программное обеспечение ЭВМ. По своему функциональному назначению все программы, входящие в состав программного обеспечения, можно разделить на три группы:

I. Программы операционной системы.

2. Программы технического обслуживания.

3. Пакеты прикладных программ.

В состав операционной системы входят управляющие и обслуживающие программы. Управляющие программы позволяют обрабатывать на ЭВМ одновременно несколько пользовательских программ, распределяют ресурсы ЭВМ, позволяют осуществлять обмен информацией между ЭВМ и внешними устройствами и т.д.

Обслуживающие программы переводят программу пользователя, написанную на языке программирования, в машинные коды, редактируют текст программы и т.д.

В группу программ технического обслуживания входят наладочные, проверочные и диагностические программы, которые позволяют инженерному персоналу, обслуживающему ЭВМ, проверять вычислительную технику и находить неисправности.

Пакеты прикладных программ (ППП) служат для максимального упрощения процедуры общения человека с ЭВМ и предназначены для автоматизации решения часто встречающихся задач. В пакеты прикладных программ входят: ППП для решения задач математики; ППП для решения задач статистики; ППП для решения задач экономики и др.

При использовании пакета прикладных программ пользователю нет необходимости программировать решение той или иной задачи. Ему достаточно знания, как обратиться к готовой программе, имеющейся в ППП, и умения совершать простейшие операции ввода исходных данных

(чисел или текстов) и вывода полученных результатов на внешнее устройство.

Программное обеспечение ПЭВМ "АГАТ"

Программное обеспечение ПЭВМ "АГАТ" состоит из следующих основных компонентов:

I. Управляющая программа "Системный монитор".

2. Файловая система.

3. Интерпретатор "Бейсик", интерпретатор "Рапира".

4. Проверочные и наладочные программы технического обслуживания.

5. Пакет прикладных программ.

Рассмотрим компоненты программного обеспечения ПЭВМ "АГАТ".

I. Программа "Системный монитор" на заводе-изготовителе записывается для долговременного хранения в ПЗУ и является ядром программного обеспечения ПЭВМ "АГАТ".

Она осуществляет:

а) запуск программ файловой системы, интерпретаторов "Бейсик" и "Рапира";

б) обмен ОЗУ с внешними устройствами (ВКУ, клавиатурой, НГМД);

в) диалог пользователя с ПЭВМ.

Запуск программы "Системный монитор" осуществляется автоматически при включении питания ПЭВМ "АГАТ".

2. Файловая система - это комплекс специальных программ, служащих для организации записи, хранения, поиска и считывания файлов.

Файлом называется порция информации (набор данных или операторов программы), хранящаяся во внешней памяти (ГМД). Для того чтобы воспользоваться необходимой обслуживающей программой, входящей в файловую систему, следует дать команду (директиву) с клавиатуры. Такие директивы будут рассмотрены при изучении темы "Работа с гибким магнитным диском".

3. Работа ЭВМ состоит в исполнении программы, состоящей из размещенных в памяти ЭВМ машинных команд. Однако создавать вручную текст на машинном языке очень сложно. Гораздо проще использовать один из языков программирования и создать программу, осуществляющую перевод с данного языка программирования на машинный язык. Для ПЭВМ "АГАТ" переводчиком являются программы интерпретатор "Бейсик" и интерпретатор "Рапира", которые читают программу пользователя по операторам, сразу выполняя перевод на машинный язык. В случае выявления ошибочных ситуаций машина выдает на экран дисплея сообщение об ошибке.

4. Проверочные и наладочные программы технического обслуживания включают в себя тесты, осуществляющие проверку правильности функционирования отдельных узлов ПЭВМ. Это тест памяти, тест языка Бейсик, тест клавиатуры и т.д.

5. Пакет прикладных программ включает в себя игровые и обучающие программы. Обучающие программы дают основные сведения по химии, информатике, алгебре и другим дисциплинам, контролируют уровень знаний.

Вопросы для повторения

1. Из каких программ состоит программное обеспечение ПЭВМ "АГАТ"?
2. Какая программа осуществляет перевод программ, написанных на языке Бейсик, на машинный язык?
3. Программа "Системный монитор" и её основное назначение.
4. Для чего нужны проверочные и наладочные программы?

3. ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА БЕЙСИК

Чтобы решить какую-либо задачу на некоторой ЭВМ, необходимо сначала продумать, как вообще можно решить эту задачу, т.е. составить алгоритм её решения (словесный, в виде графической схемы или на алгоязыке). Затем следует представить этот алгоритм в таком виде, чтобы данная ЭВМ могла его выполнить, т.е. записать данный алгоритм на каком-либо языке программирования. Такая запись называется программой для этой ЭВМ. Процесс разработки программы представляет собой процесс программирования, а человек, его выполняющий, называется программистом.

Программы для машин первого поколения записывались в машинных командах, т.е. в двоичном коде (в виде последовательности цифр "0" и "1"). Написание таких программ было очень сложно и трудоемко, да и читать такую программу было трудно.

Поэтому возникла потребность облегчения труда программиста. Был создан язык ассемблер. Этот язык использует мнемокоды. Он проще машинного языка, но всё же сложен и близок к машинному.

Первым языком высокого уровня, получившим широкое распространение, был ФОРТРАН, разработанный в 1954 г. фирмой IBM. Этот язык предназначен для решения научных задач.

Далее с развитием вычислительной техники и с расширением круга

задач, решаемых с помощью ЭВМ, появились такие языки программирования, как: Кобол, Алгол, PL/1, Бейсик и т.д.

Бейсик - это сокращенное название от Beginner's All-purpose Symbolic Instruction Code (многоцелевой язык символических команд для начинающих). Бейсик был разработан в 1965 г. в Дартмутском Университете по заказу фирмы General Electric профессорами Джоном Кемени и Томасом Куртием. Язык был создан для обучения программированию. Бейсик оставался, главным образом, учебным языком до появления в середине 70-х годов микро-ЭВМ. Бейсик был выбран для таких систем благодаря его простоте и минимальным требованиям к памяти в сравнении с другими языками высокого уровня. В настоящее время Бейсик является наиболее широко используемым в мире языком программирования для персональных ЭВМ.

Однако следует учесть, что для каждой машины существует своя версия языка, несколько отличающегося от стандартного. Эти отличия обусловлены различиями аппаратного исполнения машин.

Изучению любого языка программирования всегда предшествует знакомство с алфавитом. Выносной пульт, с помощью которого пользователь общается с вычислительной системой, - это, как правило, электрифицированная пишущая машинка или видеотерминал (выносной пульт, снабженный клавиатурой для набора сообщений и экраном для просмотра информации). Поэтому алфавит Бейсика определяется возможностями клавиатуры. В его состав входят:

1) 26 заглавных латинских букв от A до Z;

2) 10 арабских цифр от 0 до 9;

3) 4 знака препинания: . (точка); , (запятая); ; (точка с запятой); ' (апостроф);

4) 5 знаков арифметических операций: + (плюс), - (минус), * (знак умножения), /(знак деления), ∧ (знак возведения в степень);

5) 6 знаков операций отношений: = (равно), < > (не равно), > (больше), >= (больше или равно), <= (меньше или равно), < (меньше);

6) ( ) круглые скобки;

7) ␣ пробел;

8) перевод строки (↵);

9) специальные символы.

В русском варианте алфавит Бейсика расширен за счет добавления заглавных русских букв.

3.I. Представление данных

Величины делятся на переменные и константы.

3.I.I. П е р е м е н н ы е. Каждая величина имеет имя или идентификатор. Имя величины обязательно начинается с латинской буквы, затем может идти последовательность, состоящая только из латинских букв и арабских цифр. Наличие любых других символов в имени величины недопустимо.

A1, AB, MIN, SUMMA.

Величины в Бейсике бывают следующих типов: вещественные, целые и символьные. Переменные вещественного типа не помечаются отличительными символами. Для указания целых и символьных величин к их именам присовокупляются символы % и $ соответственно.

Пример:

IM - переменная вещественного типа.

SUMMA% - переменная целого типа.

OTVET $ - переменная символьного типа.

3.I.2. К о н с т а н т ы. Различают вещественные, целые и символьные константы. Вещественные константы представляются в естественном формате с фиксированной запятой:

I23.456

.53

I7684.I

или в экспоненциальной форме (с плавающей запятой)

$0.0000I234 = I.234 \cdot I0^{-5} = I.234E-5$

В этой записи латинская буква "E" сообщает ЭВМ, что для получения значения число необходимо домножить на I0 в указанной (целой) степени.

Целые константы представлены в десятичном виде: -I5 ; 2 . Числовые значения вещественных и целых переменных могут иметь не более 9 значащих цифр.

Символьные константы могут содержать от 0 до 255 любых символов, кроме [и ".

"АРБУЗ"

Если символьная константа предполагает содержание в себе ", то её нужно записать в следующем виде

"ПЭВМ - 'АГАТ' "

Вещественным и целым переменным можно присвоить только вещественную и целую константу. Символьные константы можно присвоить

только символьным переменным.

В о п р о с ы д л я п о в т о р е н и я

I. Какие символы включает в себя алфавит языка Бейсик-АГАТ?

2. Какие типы величин описывает язык Бейсик-АГАТ?

3. С какого символа начинается имя переменной? Какие и сколько символов могут входить в имя переменной?

4. Приведите примеры переменных: а) целого; б) вещественного; в) символьного типа.

5. Сколько символов могут содержать символьные константы?

6. Сколько значащих цифр могут содержать целые константы?

7. Приведите примеры констант: а) целого; б) вещественного; в) символьного типа.

3.2. Стандартные функции

При решении задач вычислительного характера без ЭВМ часто приходится прибегать к справочной литературе для нахождения значений всевозможных функций. Решая эту же задачу на ЭВМ, можно воспользоваться готовыми стандартными программами, которые хранятся в соответствующей библиотеке ЭВМ (в ПЗУ). Наиболее распространенные функции, программы вычисления которых имеются для каждой ЭВМ, принято называть стандартными.

Для обозначения стандартных функций в программе на Бейсике используются трехбуквенные имена. Аргумент функции, которым может быть произвольное арифметическое выражение, обязательно заключается в круглые скобки.

Аргументы тригонометрических функций должны задаваться в радианах. Единственная функция, не содержащая аргумента, RND работает как генератор случайных чисел из диапазона (0,I) с равномерным законом распределения.

Стандартные функции Бейсик-АГАТ представлены следующим образом:

SIN(X) - значение синуса (X рад)

COS(X) - значение косинуса (X рад)

TAN(X) - значение тангенса (X рад)

ATN(X) - значение арктангенса (X рад)

INT(X) - значение целой части X

RND - генератор случайных чисел на интервале (0,I)

SGN(X) - (-I) при X < 0; 0, при X = 0; I, при X > 0

ABS(X) - модуль X

SQR(X) – положительный квадратный корень X
EXP(X) – экспонента X
LOG(X) – натуральный логарифм X

3.3. Выражения

В программе на Бейсике допускается употребление арифметических выражений. Арифметическое выражение записывается в строку без каких-либо подстрочных или надстрочных знаков.

Математическая запись формулы	Запись на Бейсике
$\frac{ax^2 + bx + c}{d - 2,5}$	(A * X∧2 + B * X + C)/(D – 2.5)
$\sin^2 x$	SIN(X)∧2
$\sin x^2$	SIN(X∧2)

Операции выполняются слева направо в соответствии со следующей схемой предпочтения: 1) вычисления в скобках; 2) вычисление значений функций; 3) возведение в степень; 4) умножение и деление; 5) сложение и вычитание.

Вопросы для повторения

1. Что такое стандартные функции?
2. Общий вид стандартной функции.
3. Приведите примеры стандартных функций Бейсика.
4. Как записываются арифметические выражения на Бейсике?
5. Каков порядок выполнения операций в арифметическом выражении?

Упражнения

1. Укажите, какие из приведенных констант могут использоваться в Бейсике. Допущенные ошибки исправьте.

1. "ПРОГРАММА"
2. 0.25
3. "GAMMA"
4. -6,34
5. 0,34E-4
6. "1234"
7. 6.7.8.
8. -00.24
9. A x B + C
10. "765.432"
11. -0.01E/-2.5
12. "ПРАВИЛЬНО?"
13. " $\lambda \mp \beta$ "
14. МАШИНА
15. "X + 2X"
16. 37

2. Запишите для использования в Бейсике следующие числа и тексты:

1. -6,789
2. 0,25 x 10^2
3. -4,71275
4. ОТВЕТ
5. 3X + 4X
6. -РЕШЕНИЕ-
7. 1800000000
8. 0,0000014
9. б
10. дельта
11. A1234
12. BASIC

3. Какие из записанных ниже последовательностей символов могут использоваться в качестве имен переменных в Бейсике? Назовите тип переменных (цел., вещ., симв.):

1. ABC123
2. SPRINT
3. IN33%
4. I$
5. OPR(A)
6. GAMMA1
7. N%(10,24)
8. ST(K$)
9. Я10
10. BETA
11. IZ
12. %PRO
13. J
14. KL$
15. MS(I%,J%)
16. C15.6

4. Запишите по правилам Бейсика следующие выражения:

1. $\frac{x^2 - 5 + a}{5ax}$
2. $2,198 + \frac{2}{3}y$
3. $\frac{\sin 2x + \cos(x - y)}{x + y}$
4. $-\frac{1,234 + z}{z - 2,895}$
5. $\frac{(9,87 + a)(6,54 + b)}{3,21c}$
6. $\frac{(x + 2)^2}{x^2 - 1}$
7. $x^{y + z}$

4. РЕЖИМ НЕПОСРЕДСТВЕННОГО СЧЕТА

Интерпретатор Бейсика позволяет работать в двух режимах: а) непосредственного счета; б) программном.

Рассмотрим работу ПЭВМ "АГАТ" в режиме непосредственного счета. В этом режиме ПЭВМ используется как калькулятор. Предположим, что нам нужно осуществить вычисление каких-либо арифметических выражений. Для этого необходимо непосредственно после символа] , приглашающего пользователя к диалогу, набрать символ ? , затем – вычисляемое выражение на языке Бейсик и нажать на клавишу ⏎ .

? < выражение > ↵

После чего на экране ВКУ в следующей строке появляется значение выражения.

Пример I. Вычислить: 256 x 256

Р е ш е н и е. Математическое выражение запишется на Бейсике 256 * 256. Необходимо нажимать клавиши в такой последовательности:

? 256 * 256

На экране ВКУ получается ответ в следующем виде:

75536

Пример 2. Вычислить значение выражения $\frac{I25x^3 - I8}{x - I4}$ при x = 24,5.

Р е ш е н и е. Математическое выражение запишется на Бейсике:

(I25 * X∧3 - I8)/(X - I4,I)

Необходимо нажимать клавиши в такой последовательности:

? (I25 * 24.5 ∧ 3 - I8)/(24.5 - I4.I) ↵

На экране ВКУ получается ответ в следующем виде: I76754.579

Пример 3. Вычислить значение выражения cos П/16 П = 3,14.

Р е ш е н и е. Математическое выражение запишется на Бейсике:

COS (PI/16)

Необходимо нажать клавиши в такой последовательности:

? COS (3.14/16) ↵

На экране ВКУ получается ответ в следующем виде:

.980804695

При работе в режиме непосредственного счета возможны ошибочные ситуации. Например, если при наборе выражения Вы заметили, что сделали ошибку (т.е. набрали неверный символ) и клавиша ↵ ещё не нажата, необходимо, воспользовавшись клавишами управления курсором ↑ , ← , → , подвести курсор к месту ошибочно набранного символа, исправить его и далее, пробежавшись по верно набранной информации с помощью клавиши → , закончить набор выражения.

Пример 4. Вычислить значение выражения $\sqrt{\sin\frac{3П}{2}}$, где П = 3,I4.

Р е ш е н и е. На Бейсике выражение запишется в виде

SQR (SIN(3 * PI/2))

Если нажать клавиши в такой последовательности:

? SQR(SIN3 * 3.I4/2) ↵ , ответом ПЭВМ будет сообщение:

SYNTAX ERROR (синтаксическая ошибка). Это означает, что введен недопустимый на Бейсике символ. По правилам записи стандартных функций аргумент должен быть заключен в круглые скобки, т.е. следующий за именем функции SIN символ должен быть (, вместо

которого ошибочно набран аргумент 3 * 3.I4/2. Необходимо набирать строку верно:

? SQR(SIN(3 * 3.I4/2)) ↵

Ответ машины имеет вид:

ILLAGAL VALUE ERROR,

что означает: ложное значение, ошибка.

Действительно, если вычислить сначала $\sin\frac{3П}{2}$, то окажется, что в результате получим -0.999997I46, т.е. под корнем получается отрицательное значение, чего быть не должно.

Пример 5. Вычислить значение выражения $\frac{3x^2 + I5}{a^2 - I}$ при x = 5, a = I.

Р е ш е н и е. На Бейсике выражение имеет вид:

(3 * X ∧ 2 + I5)/(A ∧ 2 - I)

Подставляя значения A и X, необходимо нажимать клавиши в такой последовательности:

? (3 * 5 ∧ 2 + I5)/(I ∧ 2 - I) ↵

Ответ машины имеет вид:

DIVIZION BY ZERO,

что означает, что осуществлено деление на 0, а это невозможно, т.е. при заданных значениях A и X получилось некорректное математическое выражение, которое машина вычислить не может.

Возможен случай, когда вычисляется громоздкое математическое выражение, включающее количество символов, превышающее 256. В этом случае после нажатия клавиши ↵ ответ машины будет следующим:

LONG STRING ERROR,

что означает: длина строки превышает допустимое значение, ошибка.

Пример 6. Вычислить значение выражения [ctg ⌈2,75⌉] .

Р е ш е н и е. Вычислить значение ctgx на ПЭВМ "АГАТ" не представляется возможным, так как функции ctgx нет среди стандартных функций версии языка Бейсик-АГАТ. Из положения можно выйти, воспользовавшись формулой перехода $ctgx = \frac{1}{tgx}$.

Нажимая на клавиши в такой последовательности:

? INT(1/TAN(SQR(2.75))) ↵ ,

получим ответ: -1.

В о п р о с ы д л я п о в т о р е н и я

I. Какие режимы работы предусматривает интерпретатор языка Бейсик-АГАТ?

2. Каким образом осуществляется работа в режиме непосредственного счета?

3. Какие возможны ошибочные ситуации?

Упражнения

I. Найти при помощи ПЭВМ (в режиме непосредственного счета) ошибки в записи следующих констант на Бейсике:

7,6	8,4 E - I5
63.4 - E 2	-.400000055E3
5.06786445 E2	8.04E-I.8

2. Вычислить в режиме непосредственного счета значения приведенных выражений:

а) $7{,}5 \cdot I0^{-2} + a - 4/d$;

б) $6 + x - \frac{4{,}I}{2 + f}$;

в) $(9{,}3 + a)\cdot(3{,}2 - b)$;

г) $\frac{x^2 - x + 2a}{x^4 - 5x^2 + 20}$ при x = 78, d = 2.5, a = I5,4, b = .56E-3, f = 9

5. ДИРЕКТИВЫ ЯЗЫКА БЕЙСИК-АГАТ

Все распоряжения (приказы), которые даются пользователем ЭВМ, пишутся в виде определенным образом построенных фраз языка Бейсик-АГАТ. Фразы бывают двух типов: директивы и операторы. Директива может быть введена с клавиатуры без предшествующего ей номера, и сразу же после нажатия на клавишу перевод строки выполняется.

Рассмотрим основные директивы.

Директива NEW стирает всю находящуюся в оперативной памяти информацию.

Директива HOME очищает экран ВКУ (дисплея). При этом после выполнения директивы символы приглашения к диалогу]_ появляются в правом верхнем углу экрана. Если их нужно вернуть в нижний левый угол, необходимо нажать одновременно клавиши УПР и СБР (т.е. осуществить общий сброс).

Программы могут быть введены в память машины с магнитного диска или с клавиатуры. Строки программы должны иметь номера, в соответствии с которыми они хранятся в памяти и выполняются.

Директива LIST выводит на экран дисплея текст программы, которая хранится в оперативной памяти, начиная со строки с минимальным

номером. Существуют модификации директивы LIST.

Если требуется вывести на экран дисплея только одну строку программы с номером N , директива должна иметь вид: LIST N

Если требуется вывести на экран строки программы, начиная со строки с номером N , директива должна иметь вид: LIST N,

Если требуется вывести на экран строки программы, начиная с номера N и кончая номером M, директива должна иметь вид: LIST N,M

Директива DEL N,M удаляет из текста программы, которая хранится в оперативной памяти ПЭВМ, строки с номерами от N до M включительно.

Если требуется удалить из текста программы одну строку с номером N, директива должна иметь вид: DEL N,N

Директива RUN запускает на выполнение программу, которая хранится в оперативной памяти ПЭВМ, начиная с минимального номера.

Если требуется запустить на выполнение программу, которая хранится в памяти с номера N , директива должна иметь вид: RUN N

Работа с внешними устройствами

К внешним устройствам ПЭВМ "АГАТ" относятся дисплей (ВКУ), блок клавиатуры, накопитель на гибких магнитных дисках (НГМД) и принтер. Рассмотрим работу с гибким магнитным диском.

I. Чтение каталога программ. Для чтения каталога (перечня) имен программ (файлов), которые хранятся на гибких магнитных дисках, нужно: а) вставить ГМД в гнездо накопителя; б) дать машине директиву CATALOG. После этого на экран дисплея выводится перечень программ пользователя, хранящихся на ГМД, например:

```
A 033  KLAWIATURA
B 003  ОРГАН
.................
 T    017   СПИСОК
 1     2      3
.................
```

где 1 - тип файла (A, B, T); 2 - размер файла в секторах (1 сектор = 256 байт); 3 - имя файла.

A файл - программа, написанная на Бейсике. B файл - программа, написанная двоичными кодами. T файл - текстовая программа.

Имя файла может содержать буквы латинского или русского алфавитов и цифры, причем первой должна быть буква. Использование других символов в записи имен файлов - запрещено. Длина имени не должна превышать 250 символов.

2. Загрузка программы с ГМД в память ПЭВМ. Для загрузки программы с ГМД в оперативную память ПЭВМ нужно: а) вставить ГМД с нужной программой в НГМД; б) дать машине директиву: LOAD < имя файла >.

После окончания нормальной (чтение без сбоев) загрузки на экране ВКУ появится приглашение пользователя к работе]_ . При этом прежнее содержимое памяти стирается. В случае, если данная программа читается со сбоем, машина выдает ситуацию I/O ERROR , что означает ошибка ввода/вывода.

3. Выполнение программы с ГМД. Для непосредственного выполнения программы с магнитного диска нужно: а) вставить ГМД с нужной программой в НГМД; б) дать машине директиву: RUN < имя файла >. После этого программа (с указанным именем) загружается в память ПЭВМ и автоматически выполняется.

4. Запись программы на ГМД. Составленную и отлаженную пользователем программу можно записать на ГМД для долговременного хранения. Для записи на диск, где уже есть записанные программы, нужно:
а) ввести программу в память ПЭВМ, набирая её на клавиатуре;
б) вставить ГМД, на который нужно записать программу, в НГМД;
в) дать машине директиву: SAVE < имя файла >. После успешной записи на экране ВКУ появляются символы]_ . При этом содержимое памяти не стирается.

5. Уничтожение файла с магнитного диска. Если необходимо стереть текст программы (файл) с ГМД, нужно воспользоваться директивой: DELETE < имя файла > .

Вопросы для повторения

I. Перечислите основные директивы языка Бейсик-АГАТ и объясните их назначение.

2. Как можно убедиться в том, что нужная для работы программа хранится на предложенном диске?

3. Как загрузить программу с ГМД в память ПЭВМ?

4. Как осуществляется запуск программы на выполнение с ГМД?

5. По какой директиве осуществляется запись программы на ГМД?

Упражнения

I. Прочитайте перечень программ, хранящихся на предложенном диске.

2. Загрузите указанную программу с диска в память машины.

3. Запустите программу, находящуюся в памяти машины, на выполнение.

4. Выведите на экран дисплея текст программы, находящейся в памяти машины.

5. Уничтожьте в программе строки с I0 по 50. Проверьте выполнение данной Вами директивы.

6. Очистите память машины. Проверьте выполнение данной Вами директивы.

7. Запустите на выполнение программу, которая находится на диске.

8. Запишите на диск программу, находящуюся в памяти машины под другим именем. Проверьте правильность выполненной Вами директивы.

9. Подготовьте машину для работы другого пользователя, очистив память машины и экран ВКУ.

6. РЕЖИМ ПРОГРАММИРОВАНИЯ. ОБЩИЙ ВИД ПРОГРАММЫ НА БЕЙСИКЕ. ОСНОВНЫЕ ОПЕРАТОРЫ

Программирование – это составление инструкций для ЭВМ, указывающих, как решать задачу. Эти инструкции пишутся в виде определенным образом построенных фраз языка программирования. Такие фразы называются операторами, а их последовательность образует программу.

Любой оператор языка Бейсик начинается с числовой метки – номера оператора. Роль метки выполняют целые числа в диапазоне от I до 9999.

В Бейсике номер оператора выполняет две функции. Во-первых, он служит меткой оператора и может быть использован для ссылки на данный оператор (например, при передачах управления в программе). Во-вторых, он служит для упорядочения операторов в исходной программе перед её выполнением. Следует обратить внимание, что операторы нумеруются не подряд, а с интервалом. Обычно при составлении программ на Бейсике строки нумеруются через десять: I0, 20, 30... Это делается для того, чтобы при отладке программы при необходимости можно было вставить одну или несколько строк без перенумерации остальных.

Завершается оператор символом ⌐ "перевод строки". Максимальная длина строки составляет 256 символов. Максимальная длина строки экрана равна 32 символам. В том случае, если длина вводимой строки больше 32 символов, ПЭВМ "АГАТ" автоматически осуществляет перевод

курсора на следующую строку экрана.

Программа выполняется в порядке увеличения номеров строк, если нет дополнительных указаний, организуемых специальными операторами передачи управления.

6.1. Оператор комментария REM

В языке Бейсик есть одна особенность. Среди операторов программы могут встречаться строки, не вызывающие каких-либо машинных действий (например, всевозможные пояснения). Такие операторы называются невыполняемыми. К их числу принадлежит оператор REM (сокращение от английского слова REMARK - комментарий, замечание).

Общий вид оператора:

m REM <текст комментария>,

где m - номер строки,

REM - служебное слово.

Пример 1. 3Ø REM ВЫЧИСЛЯЕТСЯ ЗНАЧЕНИЕ ФУНКЦИИ

В программе можно использовать любое количество операторов REM, размещая их среди других операторов.

6.2. Оператор присваивания LET

Смысл любой программы состоит в том, чтобы в процессе её выполнения определенным переменным были присвоены те или иные значения. Для этого используется оператор присваивания LET (пусть).

Общий вид оператора:

m [LET] < X = A >,

где m - номер оператора;

LET - служебное слово (пусть). Здесь и далее в квадратных скобках указывается необязательная часть оператора;

X - имя переменной;

A - арифметическое выражение, или символьная константа.

Выполнение оператора присваивания сводится к двум последовательным шагам: вычисляется значение выражения, стоящего справа от знака равенства, и найденное значение засылается в ячейку оперативной памяти ЭВМ, выделенную для хранения указанной в операторе переменной.

Пример 2.

```
1Ø REM ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ ФУНКЦИИ
2Ø LET X = 3
3Ø LET У = ABS(X ∧ 2 - 5 * X)
```


В приведенном примере вначале вычисляется значение выражения ABS(X ∧ 2 - 5 * X), а уже затем переменной У присваивается полученное значение.

В версии Бейсик-АГАТ служебное слово LET можно опустить.

6.3. Оператор вывода PRINT

Для вывода информации на экран ВКУ в Бейсике используют оператор PRINT (печатать).

Общий вид оператора:

m PRINT <список>,

где m - номер оператора,

PRINT - служебное слово.

Элементами списка могут быть имена переменных, арифметические выражения и тексты, т.е. символьные константы. Текст используется для печати всевозможных заголовков и пояснений. Он заключается в кавычки и может состоять из произвольной последовательности символов (включая и русский алфавит). Текст не должен содержать запрещенных символов " (кавычки) и ↵ (перевод строки).

Пример 3. 1Ø PRINT "ВНИМАНИЕ!"

Результатом работы оператора с меткой 1Ø будет сообщение на экране ВКУ ВНИМАНИЕ!

Следует помнить, что завершением каждой строки программы является нажатие клавиши ↵ (перевод строки). Для исполнения данной программы машиной необходимо ввести директиву RUN.

Для того чтобы вывод текста начинался не с первой позиции строки, необходимо внести в текст пробелы.

Пример 4. Вывести текст ВНИМАНИЕ!, начиная с 5-й позиции строки

1Ø PRINT "␣␣␣␣ВНИМАНИЕ!"

Для пропуска на печати нужного числа строк используется пустой оператор PRINT.

Пример 5. Вывести текст ВНИМАНИЕ! через 3 строки с 5-й позиции.

```
1Ø PRINT
2Ø PRINT
3Ø PRINT
4Ø PRINT "␣␣␣␣ВНИМАНИЕ!"
```

Для того чтобы текст был выведен на чистом экране, необходимо в программу включить директиву HOME. Однако следует помнить, что директива HOME перемещает курсор в левый верхний угол экрана.

Пример 6.

```
1Ø  HOME
2Ø  PRINT
3Ø  PRINT
4Ø  PRINT
5Ø  PRINT "␣␣␣␣ВНИМАНИЕ!"
```

В результате выполнения данной программы текст ВНИМАНИЕ! будет расположен в четвертой сверху строке, начиная с пятой позиции.

Пример 7. Написать программу вычисления значения выражения $y = x^2 - 5x$ при x = 3.

Программа:

```
1Ø REM ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ ВЫРАЖЕНИЯ
2Ø LET X = 3
3Ø LET У = ABS(X ∧ 2 - 5 * X)
4Ø PRINT  У
```

В результате выполнения оператора 3Ø вычислится значение выражения ABS(X ∧ 2 - 5 * X) при X = 3 и У присвоится значение 6. По оператору 4Ø на экран дисплея будет выведено числовое значение У в виде: 6 .

В зависимости от того, какой разделитель используют между переменными, символьными константами и выражениями, в списке вывода оператора PRINT различают два способа вывода. Разделителями могут быть запятая или точка с запятой. Например, в предыдущей программе можно было вывести не только числовое значение У, но и имя этой переменной или какой-то текст, поясняющий полученное числовое значение результата.

Оператор 4Ø будет иметь вид: 4Ø PRINT "У = "; У

Между текстовой константой "У = " и переменной У поставлен разделитель ";" (точка с запятой), который говорит о том, что печать на экран числового значения У, равного 6, будет осуществлена вплотную, т.е. без пробела в следующей после символа = позиции. На экране дисплея оператор 4Ø напечатает У = 6.

Пример 8. Написать программу вычисления значений выражений У = ABS(X ∧ 2 + 5 * X) и $K = \sqrt{(X + 1)}$ при X = 3 и вывести результаты вычислений на экран дисплея.

Программа:

```
1Ø  REM ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ВЫРАЖЕНИЙ
2Ø  LET X = 3
3Ø  LET У = ABS(X ∧ 2 + 5 * X)
```



```
4Ø  LET K = SQR(X + 1)
5Ø  PRINT "У = "; У; "К = "; К
```

В результате работы данной программы на экране дисплея появится печать значений результатов У и К в виде: У=I2К=2

Если в качестве разделителя используется запятая, то следует знать, что строка экрана условно делится на 3 зоны по 8 позиций каждая и печать очередного результата будет производиться в следующей зоне.

Если в примере 8 в операторе 5Ø использовать разделителем запятую, т.е.

```
5Ø  PRINT "У = "; К, "К = "; К
```

то на экран дисплея выведется значение К, начиная с 9-й позиции строки:

```
У = I2        К = 2
. . . . . .
```

Можно записать оператор 5Ø в виде:

```
5Ø  PRINT  "У = "; У,, "К = "; К
```

В этом случае печать результата К переместится на одну зону (8 позиций) и результат К напечатается, начиная с I7-й позиции строки:

```
У = I2                    К = 2
. . . . . .
```

6.4. Операторы ввода INPUT и DATA-READ

Для ввода значений исходных данных используются операторы INPUT и DATA-READ.

Общий вид оператора INPUT:

```
m INPUT  ["t";]  X1, X2, ... , XN,
```

где m – номер строки;

INPUT – служебное слово;

t – символьная константа, поясняющая запрос исходных данных (наличие квадратных скобок свидетельствует о том, что это необязательная часть оператора);

X1, X2, ... , XN – список имен переменных.

Пример 9. Написать программу вычисления значения выражения У = ABS(X ∧ 2 - 5 * X) для различных значений X.

Программа:

```
IØ  REM СЧЕТ
2Ø  INPUT  X
3Ø  У = ABS(X ∧ 2 - 5 * X)
4Ø  PRINT У
```

При выполнении оператора INPUT вычисления приостанавливаются, и ПЭВМ "АГАТ" напечатает на экране дисплея знак "?". Вопрос напоминает пользователю, что с клавиатуры должно быть введено значение переменной X. После ввода переменной и нажатия на клавишу ↵ (перевод строки) машина продолжит вычисления. Если использовать символьную константу, то на экран ВКУ выводится сообщение t .

Оператор 2Ø может иметь вид:

```
2Ø  INPUT "ВВЕСТИ ЗНАЧЕНИЕ X"; X
```

В результате на экране ВКУ при выполнении оператора 2Ø появится сообщение ВВЕСТИ ЗНАЧЕНИЯ X и ЭВМ будет ждать ввода значений переменной X.

Текст, поясняющий ввод переменной, можно вывести на экран ВКУ оператором PRINT. Программа примера 9 может иметь вид:

```
IØ  REM СЧЕТ
2Ø  PRINT "ВВЕСТИ ЗНАЧЕНИЕ X"
3Ø  INPUT X
4Ø  У = SQR(X ∧ 2 - 5 * X)
5Ø  PRINT У
```

Следует обратить внимание, что в примере 9 оператором INPUT вводится значение одной переменной. Их может быть любое количество.

Пример IO. Написать программу вычисления значения выражения $У = X^3 + C$ для различных значений X и C.

Программа:

```
IØ  REM СЧЕТ
2Ø  INPUT "ВВЕСТИ ЗНАЧЕНИЯ X, C"; X, C
3Ø  LET У = X ∧ 3 + C
4Ø  PRINT "У = "; У
```

При выполнении оператора 2Ø вычисления приостанавливаются, и машина напечатает на экране знак "?", по которому необходимо ввести числовые значения переменных X и C.

Если в операторе INPUT предусмотрен ввод трех переменных, а введено только две переменные, то на экране ВКУ появятся два знака вопроса (??). Выполнение программы будет продолжено только после ввода значения третьей переменной.

Общий вид оператора DATA-READ:


```
m READ  X1, X2, ... , XN
. . . . . .
n DATA  C1, C2, ... , CN ,
```

где m, n- номера строк;

X1, X2, ... , XN - список имен переменных;

C1, C2, ... , CN - значения, соответствующие именам переменных, указанных в операторе READ .

В исходной программе может быть и несколько операторов DATA. Оператор DATA принадлежит к группе невыполняемых операторов, поэтому он может стоять в любом месте программы.

Оператор DATA создает блок данных из констант C1, C2, ... , CN. Оператор READ выбирает из блока данных, сформированного оператором DATA, конкретные значения и присваивает их соответствующим переменным X1, X2, ... , XN.

Пример II. Вычислить значение выражения $У = AB - C^2$ при A = 0,7; B = -8; C = 9. Ввод исходных данных осуществить с помощью операторов DATA-READ .

```
IØ  REM СЧЕТ
2Ø  READ A, B, C
3Ø  У = A * B - C ∧ 2
4Ø  DATA Ø.7, -8, 9
5Ø  PRINT У
```

В этом случае в результате работы оператора READ переменным A, B, C будут присвоены значения A = 0.7 B = -8 C = 9 .

К одному оператору DATA могут обращаться несколько операторов READ.

Например:

```
IØ  REM СЧЕТ
2Ø  READ A, B, C
3Ø  READ  D, E, F
4Ø  У = A * B * C - D * E * F
5Ø  PRINT У
6Ø  DATA Ø.7, -8, 9, -I.4, 7.3, -Ø.43
```

В этом случае в результате работы 1-го оператора READ переменным A, B, C будут присвоены значения A = 0.7 B = -8 C = 9 .

2-й оператор READ осуществляет следующие присваивания:
D = -I.4 E = 7.3 F = -Ø.43 .

В том случае, если количество переменных в операторах READ превышает количество данных в операторах DATA , что неверно, на экране дисплея появится сообщение об ошибке:

NO DATA ERROR - НЕТ ДАННЫХ, ошибка

Операторы READ-DATA так же, как и оператор INPUT, используются для ввода данных в программу. Различие состоит в том, что при использовании оператора READ все данные должны быть заранее включены в программу в операторах DATA и выборка значений переменных осуществляется автоматически. При использовании оператора INPUT вычисления приостанавливаются и возобновляются только после ввода значений запрашиваемых переменных.

6.5. Оператор END

Оператор END служит признаком конца программы, поэтому он всегда должен иметь максимальный номер и стоять в конце программы.

Общий вид оператора: m END , где m - номер строки; END - служебное слово (конец).

Вопросы для повторения

I. С помощью какого оператора вводятся пояснения в программе? Общий вид оператора.

2. Назначение и общий вид оператора присваивания.

3. С помощью каких операторов можно ввести исходные данные?

4. Где может быть расположен оператор DATA?

5. Какое сообщение напечатает ЭВМ при попытке читать данные из оператора DATA , когда в нем не осталось ни одного значения?

6. Общий вид оператора INPUT. Работа оператора.

7. Каким образом ЭВМ сообщает, что ей надо ввести исходные данные?

8. Оператор вывода данных. Работа оператора.

Упражнения

I. Присвойте переменной А значение 35. Что при этом происходит?

2. Как понимать оператор присваивания 2Ø LET B = B + 4 ?

3. Выведите на печать следующее сообщение: /СУММА ПОЛОЖИТЕЛЬНЫХ ЭЛЕМЕНТОВ/.

4. Выведите на печать значение переменной У.

5. Введите с помощью оператора INPUT переменные X, K.

6. Осуществите ввод переменных L и F с использованием текстовой константы оператора INPUT. Текст константы может быть следующий: "ВВЕДИТЕ ДЛИНУ И ШАГ".

7. Сформируйте блок данных из чисел 7; 0,5; -3,4; 6,2.

8. Осуществите выборку из блока данных чисел так, чтобы переменным А, В, С, Е были присвоены значения из блока данных (п.7).

7. РЕАЛИЗАЦИЯ НА БЕЙСИКЕ РАЗВЕТВЛЕНИЙ

7.I. Оператор условного перехода

Нумерация строк в программе на Бейсике определяет естественный порядок выполнения операторов.

Однако решение многих задач редко укладывается в простую линейную схему. Очень часто встречаются такие задачи, решение которых зависит от проверки какого-либо условия. В курсе информатики IX класса алгоритмы решения подобных задач уже рассматривались и назывались разветвляющимися.

Команда ветвления имеет следующий вид:

если условие
    то серия I
    иначе серия 2
всё

В Бейсике для реализации этой команды используется оператор условного перехода.

Общий вид его таков:

m IF < условие > THEN { метка / оператор / операторы ,

где m - номер строки.

В условном операторе используются служебные слова

IF - если, THEN - то .

Условие, которое в операторе записывается между служебными словами IF и THEN, представляет собой проверку истинности одного из отношений:

< - меньше
<= - меньше или равно
> - больше
>= - больше или равно
= - равно
<> - не равно

В простых условиях можно сравнивать не только константы или значения переменных, но и значения математических выражений,

например:

1. A + B > S + 1
2. B > = 100
3. B ^ 2 - 4 * A * C > = 0

Допустимо использование составных условий, при записи которых простые условия соединяются между собой служебными словами:

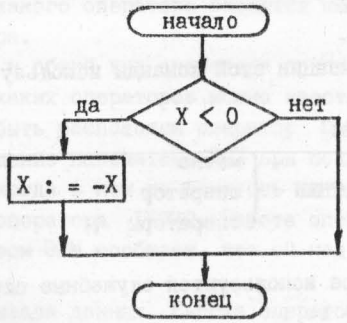
AND - и, OR - или, NOT - не. Например:

1. A > X AND X > = B
2. NOT A > B
3. K1$ = "H2S04" OR K2$ = "H2S04"

Справа от служебного слова THEN может быть записан оператор, или несколько операторов, или номер строки, на которую будет передано управление в случае соблюдения условия. Если условие не соблюдается, то выполняется оператор, стоящий в строке, следующей за оператором IF. Рассмотрим примеры.

Пример 1. Написать программу нахождения модуля числа X.

Графическая схема решения данной задачи имеет вид:



Программа:

```
10 REM МОДУЛЬ
20 INPUT X
30 IF X < 0 THEN LET X = -X
40 PRINT X
50 END
```

В случае соблюдения условия X < 0, стоящего за служебным словом IF в 30 строке, выполняется оператор присваивания LET X = -X. Затем выполняется оператор, стоящий под номером 40, т.е. осуществляется вывод результата. Если же условие не соблюдается (по оператору INPUT пользователь вводит положительное число), оператор,

следующий за служебным словом THEN, не выполняется, а осуществляется вывод результата по оператору 40 строки.

Пример 2. Даны формулы трех кислот. Определить, есть ли среди них формула серной кислоты.

Р е ш е н и е:

```
алг Химия (лит K1, K2, K3, S)
    арг K1, K2, K3
    рез
нач
    S: = "нет"
    если K1 = "H2SO4" или K2 = "H2SO4" или K3 = "H2SO4"
        то S: = "есть"
    все
кон
```

При записи данного алгоритма на алгоритмическом языке была использована сокращенная форма составной команды ветвления:

```
если условие
    то серия команд
все
```

Запишем данный алгоритм решения задачи на языке Бейсик:

```
10 REM ПОИСК СЕРНОЙ КИСЛОТЫ
20 INPUT K1$ , K2$ , K3$
30 LET S$ = "НЕТ"
40 IF K1$ = "H2SO4" OR K2$ = "H2SO4" OR K3$ = "H2SO4"
   THEN S$ = "ЕСТЬ"
50 PRINT S$
60 END
```

В случае, если составное условие в операторе IF окажется верным, то будет выполнен оператор, стоящий справа от служебного слова THEN , и переменная S $ получит значение "есть", в противном случае будет выполнена строка номер 50 и на экран будет выведено значение переменной S$, полученное при выполнении строки номер 30.

7.2. Оператор безусловного перехода GOTO

С помощью оператора GOTO можно передавать управление на любую строку программы.

Общая форма записи оператора:

m GOTO n (идти к),

где m - номер строки;

n - номер строки, которой передается управление в результате выполнения оператора GOTO, т.е. следующей будет выполняться строка с номером n . Например:

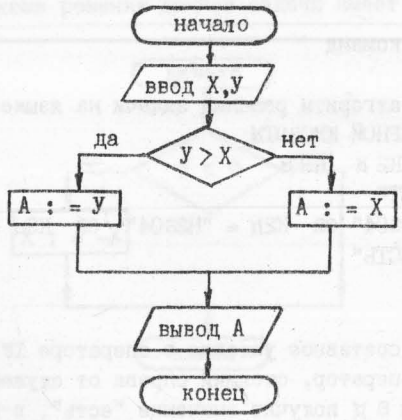
```
.....
2Ø LET MAX = A
3Ø GOTO 5Ø
4Ø LET M = I + K
5Ø PRINT A
.....
```

В этом фрагменте программы после строки с номером 3Ø выполнится строка с номером 5Ø.

Рассмотрим примеры.

Пример I. Присвоить переменной А то значение, которое имеет бо́льшая из величин Х и У. Если Х = У, то присвоить переменной А значение У.

Запишем алгоритм с помощью графической схемы:




```
IØ REM НАХОЖДЕНИЕ НАИБОЛЬШЕГО ИЗ ДВУХ ЧИСЕЛ Х И У
2Ø INPUT X, У
3Ø IF У >= X THEN 6Ø
4Ø LET A = X
5Ø GOTO 7Ø
6Ø LET A = У
7Ø PRINT "A = "; A
8Ø END
```


Если условие У >= Х выполняется, то управление передается на строку программы с номером 6Ø, в случае невыполнения данного условия естественный порядок выполнения строк программы сохраняется, т.е. величине А будет присвоено значение переменной Х. Оператор безусловного перехода GOTO 7Ø позволяет разделить ветви составной команды ветвления и передать управление со строки номер 5Ø на строку с номером 7Ø.

Пример 2. Написать программу решения квадратного уравнения $ax^2 + bx + c = 0$.

```
IØ REM РЕШЕНИЕ КВАДРАТНОГО УРАВНЕНИЯ
2Ø PRINT "ВВЕДИТЕ КОЭФФИЦИЕНТЫ A, B, C"
3Ø INPUT A, B, C
4Ø LET D = B ^ 2 - 4 * A * C
5Ø IF D >= Ø THEN 8Ø
6Ø PRINT "УРАВНЕНИЕ РЕШЕНИЙ НЕ ИМЕЕТ"
7Ø GOTO I2Ø
8Ø D = SQR(D)
9Ø XI = (-B + D)/(2 * A)
IØØ X2 = (-B - D)/(2 * A)
IIØ PRINT "X1 = "; X1, "X2 = "; X2
I2Ø END
```

Пример 3. Составить программу вычисления значения функции

$$y = \begin{cases} a - x, & \text{если } x < 0 \\ x, & \text{если } x > 0 \\ a, & \text{если } x = 0 \end{cases}$$

Для решения этой задачи удобнее воспользоваться командой выбора.

Алгоритм решения данной задачи на алгоритмическом языке выглядит следующим образом:

```
алг  ВЗФ (вещ x, a, y)
     арг  a, x
     рез  y
нач
     выбор
          при x < 0: y:= a - x
          при x > 0: y:=  x
          иначе  y:= a
     всё
кон
```

В одной строке программы может содержаться несколько операторов, они разделяются символом ":" (двоеточие).

Программа на Бейсике имеет вид:

```
1Ø REM ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ ФУНКЦИИ
2Ø INPUT "ВВЕСТИ ЗНАЧЕНИЯ X, A"; X, A
3Ø IF X < Ø  THEN У = A - X: GOTO 6Ø
4Ø IF X > Ø  THEN У = X: GOTO 6Ø
5Ø У = A
6Ø PRINT "У ="; У
7Ø END
```

Вопросы для повторения

1. Когда используется оператор условного перехода?
2. Каков общий вид оператора условного перехода?
3. Объясните работу оператора IF.
4. Что такое безусловная передача управления? Какой оператор используется для этого в Бейсике?

Упражнения

1. Объясните, как сработает следующий оператор при разных значениях I: 2Ø IF I>Ø THEN I2Ø .
2. Напишите программу, которая вводит с клавиатуры два числа и выводит на экран квадрат большего из них.
3. Запишите на Бейсике следующие фрагменты алгоритмов:

а) если $A > B$
то $У := A^2 + B$
иначе $У := B^2 + A$
всё

б) если $X < A$ и $X > B$
то $У :=$ "$B < X < A$"
всё

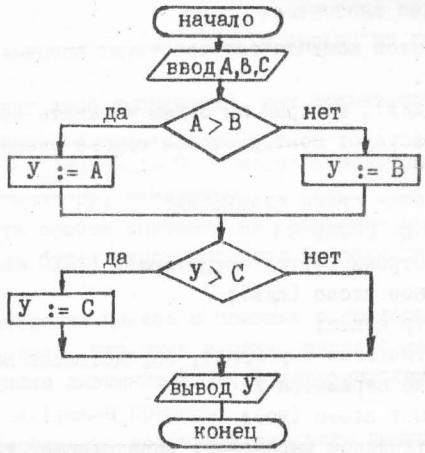
4. Составьте программы решения следующих задач:

а) найти меньшее из двух чисел;

б) решить линейное уравнение (ЛУР) $AX = B$, где A и B – произвольные вещественные числа;

в) решить неравенство $AX > B$ (НЕР), где A и B – произвольные вещественные числа.

5. Дан алгоритм, описанный с помощью графической схемы:



Сформулируйте условие задачи и напишите программу на Бейсике.

6. Дана программа:

```
1Ø REM. . .
2Ø INPUT "ВВЕСТИ ЗНАЧЕНИЕ X"; X
3Ø IF X < Ø THEN 6Ø
4Ø LET У = X
5Ø GOTO 7Ø
6Ø LET У = -X
7Ø PRINT "У = "; У
8Ø END
```

Сформулируйте условие задачи, решение которой описано данной программой, и впишите комментарий.

8. ЦИКЛЫ

Циклы – это участки программы, которые в процессе её выполнения повторяются многократно. Используя циклы, мы можем писать сравнительно короткие программы, по которым ЭВМ будет выполнять значительный объем вычислений.

В алгоритмическом языке школьного курса информатики используется команда цикла – команда повторения с параметром:

для X от A до B шаг C

```
нц
    серия команд
кц
```

В языке Бейсик этой команде соответствуют специальные операторы цикла FOR и NEXT .

Оператор FOR (для), который мы будем называть заголовком цикла, всегда предшествует повторяющейся группе операторов, составляющих тело цикла.

Общий вид заголовка цикла следующий:

m FOR X = A TO B [STEP C]

Здесь m – номер строки (метка оператора);
FOR – служебное слово (для);
X – параметр цикла;
A – арифметическое выражение, определяющее начальное значение параметра X;
TO – служебное слово (до);
B – арифметическое выражение, определяющее конечное значение параметра X;
STEP – служебное слово (шаг);
C – арифметическое выражение, определяющее величину приращения (шаг) параметра.

Если STEP C не указан, приращение параметра цикла считается равным 1.

Во время выполнения оператора FOR вычисляются начальное и конечное значения параметра X и начальное значение присваивается параметру.

Тело цикла начинается вслед за оператором FOR и завершается оператором NEXT (следующий).

Оператор NEXT записывается в следующем виде:

m NEXT [X]

Здесь m – номер строки (метка оператора);
NEXT – служебное слово (следующий), конец цикла;
X – имя параметра цикла.

При выполнении оператора NEXT изменяется значение параметра (X : = X + C) и производится анализ на конец цикла.

Например:

Фрагмент программы:	Пояснения:
. . .	
2Ø FOR I = 1 TO IØØ STEP2	заголовок цикла

3Ø K = K + 2 * I 4Ø PRINT "K = "; K	тело цикла
5Ø NEXT I	увеличение параметра цикла и анализ на конец цикла

. . .

Цикл прекратит свое выполнение, как только параметр цикла станет строго больше (при положительном шаге) или строго меньше (при отрицательном шаге) значения B. В этом случае управление в программе передается оператору, следующему за NEXT.

Надо обратить особое внимание на следующие правила:

1. Тело цикла будет выполнено хотя бы один раз при любых значениях A, B и C.

2. При организации циклов с помощью операторов FOR и NEXT следует иметь в виду, что вход в цикл, минуя оператор FOR, запрещен.

3. В теле цикла значения A, B, C перевычисляться не могут.

Рассмотрим следующие примеры.

Пример 1. Необходимо составить алгоритм вычисления $y = n!$, где $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n - 1) \cdot n$

Запишем алгоритм на алгоритмическом языке:

```
алг  ФАКТОРИАЛ (нат n, y)
     арг  n
     рез  y
нач  нат  i
     y : = 1
     для i от 1 до n шаг 1
     нц
        y : = y • i
     кц
кон
```

На языке программирования Бейсик алгоритм решения этой задачи будет выглядеть так:

```
1Ø REM ФАКТОРИАЛ
2Ø PRINT "ВВЕСТИ НАТУРАЛЬНОЕ ЧИСЛО  "
3Ø INPUT N
4Ø Y = 1
5Ø REM ДАЛЕЕ СЛЕДУЕТ ЦИКЛ ВЫЧИСЛЕНИЯ Y
6Ø FOR I = 1 TO N
7Ø Y = Y * I
8Ø NEXT I
```

```
9Ø REM ВЫВОД НА ПЕЧАТЬ РЕЗУЛЬТАТА
IØØ PRINT "У = "; У
IIØ END
```

В этой программе через I обозначен текущий элемент натурального ряда. У – это переменная, в которой будем накапливать произведение натуральных чисел, следовательно, вначале необходимо сделать эту переменную равной 1 (оператор 4Ø). Операторы 6Ø, 7Ø, 8Ø реализуют цикл, в котором будет накапливаться произведение членов натурального ряда (факториал натурального числа).

Пример 2. Составить алгоритм вычисления суммы

$$\frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{6^2} + \ldots + \frac{1}{I28^2}.$$

На алгоритмическом языке:

```
алг  СУММА  (вещ  S)
     рез  S
нач  нат  i
     S : = 0
     для  i  от  2  до  I28  шаг  2
     нц
        S : = S + 1/i^2
     кц
кон
```

На языке Бейсик:

```
IØ REM СУММА
2Ø S = 0
3Ø FOR I = 2 TO I28 STEP 2
4Ø S = S + 1/I ∧ 2
5Ø NEXT I
6Ø PRINT "S = "; S
7Ø END
```

Оператор 2Ø – очистка нулём переменной S, в которой будет накапливаться сумма $\frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{6^2} + \ldots + \frac{1}{I28^2}$.

Оператор с меткой 3Ø – заголовок цикла, в котором задаются начальное значение текущего элемента последовательности (I = 2), конечное значение (I28), а также величина приращения (2).

В теле цикла (оператор 4Ø) описана работа, которую нужно проделать для каждого допустимого значения параметра цикла I.

В операторе 5Ø NEXT I происходит изменение значения I на величину шага, т.е. на 2 (I = I + 2), а также сравнение с конечным значением (I28). Таким образом, оператор 4Ø (тело цикла) будет выполняться многократно, пока I ⩽ I28. Именно в результате работы операторов FOR-NEXT в переменной S будет получена требуемая сумма.

Вопросы для повторения

1. Для чего служат операторы FOR и NEXT?
2. Для чего в операторе FOR задается шаг (STEP)? Что произойдет, если программист не укажет величину шага?
3. Какова функция оператора NEXT в цикле.
4. Куда передается управление в программе после завершения FOR-NEXT цикла?

Упражнения

1. Проанализируйте оператор FOR :

```
. . .
3ØØ FOR K = 1 TO 36 STEP 3
. . .
5ØØ NEXT K
. . .
```

Какое значение будет иметь параметр цикла K по окончании цикла?

2. Что напечатает программа:

```
IØ  A = Ø
2Ø FOR L = IØØ  TO  I5  STEP -IØ
3Ø  A = A + L
4Ø NEXT L
5Ø PRINT A
```

3. Что вычисляет следующий алгоритм:

```
алг  СУММА  (вещ  S)
     рез  S
нач  нат  m
     S : = Ø
     для  m  от  1  до  5Ø  шаг  1
     нц
        S : = S + (m + 1)^2 / (m + sin m)
     кц
кон
```

Составьте программу, соответствующую этому алгоритму.

4. Напишите программу вычисления

$$S = 1^2 + 2^2 + 3^2 + \ldots + 100^2 .$$

5. Напишите программу вычисления всех значений функции

$$y = \begin{cases} 3x^2 + 1, & \text{если } x \geqslant 0 \\ 2x^2 - 5, & \text{если } x < 0 \end{cases}$$

для x = -10, -9, -8, ... , 0, 1, 2, 3, ... , 10

Предусмотреть печать каждого полученного значения y.

6. Составьте программу вычисления значения функции

$$F(x) = \frac{(x-2)\cdot(x-4)\cdot(x-6)\cdot \ldots \cdot(x-64)}{(x-1)\cdot(x-3)\cdot(x-5)\cdot \ldots \cdot(x-63)} .$$

9. МАССИВЫ

В большинстве практических задач, связанных с обработкой информации, приходится иметь дело с конечными числовыми и нечисловыми последовательностями, образование членов которых не подчиняется какой-либо зависимости. Элементы каждой такой последовательности могут быть обозначены одним и тем же именем с различными индексами. Такие совокупности называются массивами (таблицами).

В Бейсик-АГАТЕ могут быть использованы одномерные массивы (линейные таблицы) и двумерные массивы (прямоугольные таблицы или матрицы).

В одномерном массиве каждый элемент снабжается одним индексом - порядковым номером элемента в массиве.

Положение каждого элемента в двумерном массиве определяется номером строки и номером столбца, на пересечении которых он расположен. Следовательно, каждый элемент двумерного массива снабжается двумя индексами.

Минимальное значение индексов равно 0. При записи элемента массива индексы (индекс) записываются в круглых скобках после имени массива. Имя массива - последовательность латинских букв и цифр, начинающаяся с буквы. Например, B (10) - десятый по порядку элемент массива B, состоящего из вещественных чисел, A$(3) - третий по порядку элемент символьного массива A$, MAS(I) - элемент вещественного массива MAS с порядковым номером I (текущий элемент), C (3, 4) - элемент двумерного вещественного массива C, стоящий на пересечении 3-й строки и 4-го столбца.

Следует обратить внимание на то, что тип элементов, входящих

в массив, описывается по тем же правилам, что и простые переменные (см. разд.3).

Массив имеет не только тип, но и размер, поэтому в программе он обязательно должен быть описан. В программах это делается с помощью оператора DIM (DIMENSION - размерность):

m DIM $A_1(M_1)$, $A_2(M_2)$, ... , $A_n(M_n)$,

где m - номер строки (метка оператора);

A_i - имя массива (последовательность латинских букв и цифр, начинающаяся с буквы);

M_i - границы массива A_i (одно или два целых числа, разделенные запятыми и указывающие максимальное значение индексов).

Например, оператор 10 DIM A%(5), C(2,3) описывает два массива: целый массив A, состоящий из 6 элементов (A%(0), A%(1), A%(2), A%(3), A%(4), A%(5)), и двумерный вещественный массив C, содержащий 3 строки и 4 столбца (т.е. 12 элементов).

Оператор DIM должен располагаться в начале программы, до операторов, работающих с элементами массивов.

По описанию массива в памяти выделяется место для размещения его элементов.

Для работы с массивом необходимо иметь элементы массива в оперативной памяти ПЭВМ.

Для ввода значений элементов массивов в память можно использовать оператор INPUT .

Таким образом, если в программе описан одномерный массив A в операторе DIM A(4), то для ввода значений его элементов можно написать 10 INPUT A(0), A(1), A(2), A(3), A(4) .

Работа за дисплеем будет идти так:

```
? 2.25, 10.3, 8.6, 0.15, 9.9 ↵
```

Элементы массива в оперативную память можно вводить также с помощью операторов DATA-READ .

Пример использования операторов READ и DATA для ввода элементов описанного выше массива A:

```
10 DATA 2.25, 10.3, 8.6, 0.15, 9.9
. . .
60 READ A(0), A(1), A(2), A(3), A(4)
```

В операторе DATA константы "хранятся", а вводятся оператором READ в соответствующие переменные.

В результате работы приведенных операторов переменным будут присвоены следующие значения:

A(0) = 2.25, A(1) = 10.3, A(2) = 8.6, A(3) = 0.15, A(4) = 9.9 .

В чем разница между двумя описанными способами ввода? В случае оператора INPUT значения для переменных вводятся с клавиатуры. Выполнение программы прекращается, и ЭВМ ждет, когда соответствующее значение будет набрано на клавиатуре. Во втором случае остановка программы не происходит, так как значения вводимых данных имеются в самой программе. ЭВМ находит соответствующий оператор DATA и из него берет необходимые значения. Если в имеющихся операторах DATA не хватает констант для переменных, указанных в операторе READ, ЭВМ напечатает на экране сообщение об ошибке:

NO DATA ERROR .

Когда массив содержит большое число элементов, неудобно перечислять их имена (т.е. переменные с индексами) в операторах ввода. Гораздо удобнее воспользоваться оператором цикла.

Ниже приведены фрагменты программы, организующие ввод элементов одномерного массива:

а)
```
10 DIM  A(10)
20 FOR I = 0  TO  10
30 INPUT  A(I)
40 NEXT I
```

б)
```
10 DIM  A(10)
20 DATA  2, 4, 6, 5, 3.2, 10, 25.2, 13.8, 20.4, I, I5
30 FOR I = 0  TO  10
40 READ A(I)
50 NEXT I
```

В варианте а) программа 11 раз высветит ? для ввода всех значений элементов массива A, которые надо будет последовательно набрать на клавиатуре.

В варианте б) оператор READ A(I) прочитает эти 11 значений из оператора DATA .

Ввод двумерных массивов (прямоугольных таблиц) потребует использования вложенных циклов.

Ниже приведен пример ввода в машину целого двумерного массива C% из 4 строк и 5 столбцов:

```
10 DIM C%(4,5)
20 FOR I = 0  TO 4
30 FOR J = 0  TO 5
40 INPUT  C%(I,J)
50 NEXT J
60 NEXT I
```


Более естественно нумеровать элементы массива, начиная не с нулевого, а с первого, поэтому нами обычно игнорируются элементы массивов с индексом 0.

Рассмотрим несколько примеров.

Пример 1. Найти сумму всех элементов одномерного массива A, состоящего из 10 чисел.

На алгоритмическом языке данный алгоритм имеет вид:

```
алг  СУММА  (вещ таб  A [1 : 10]  , вещ S )
    арг  A
    рез  S
нач цел  i
    S : = 0
    для  i  от  1  до  10
        нц
        S : = S + A [i]
        кц
кон
```

Программа на Бейсике:	Пояснения :
`10 REM СУММА`	
`20 DIM A(10)`	описание вещественного одномерного массива A из 10 чисел

`30 FOR I = 1 TO  10`
`40 INPUT A(I)`
`50 NEXT I`
	цикл ввода элементов массива A в память

`60  S = 0`	очистка переменной S , в которой будем накапливать сумму

`70 FOR I = 1  TO  10`
`80 S = S + A(I)`
`90 NEXT I`
	цикл, в котором происходит суммирование каждого очередного элемента массива A в переменной S

`100 PRINT "СУММА = " ; S`	печать результата
`110 END`	

Пример 2. В одномерном целочисленном массиве, состоящем из N чисел, подсчитать количество положительных элементов.

Программа:	Пояснения
`10 REM ПОДСЧЕТ`	
`20 PRINT "ВВЕДИТЕ РАЗМЕРНОСТЬ МАССИВА"`	
`30 INPUT N`	
`40 DIM B%(N)`	
`50 K% = 0`	

```
6Ø FOR I = 1 TO N
7Ø INPUT B% (I)
8Ø IF B%(I)< Ø THEN  K% = K% + 1
9Ø NEXT I
```
} цикл, в котором осуществляется ввод элементов массива B% в память и подсчет количества отрицательных чисел в массиве

```
IØØ PRINT "КОЛИЧЕСТВО ПОЛОЖИТЕЛЬНЫХ ЧИСЕЛ = "; K
```

Пример 3. Составить программу замены нулевых элементов одномерного массива единицами.

Программа: Пояснения

```
 IØ REM  ЗАМЕНА
 2Ø PRINT "РАЗМЕРНОСТЬ МАССИВА"
 3Ø INPUT N
 4Ø DIM  C(N)
 5Ø FOR I = 1 TO N
 6Ø INPUT C(I)
 7Ø NEXT I
```
} ввод элементов массива в память

```
 8Ø FOR I = 1 TO N
 9Ø IF C(I) = Ø THEN  C(I) = 1
IØØ PRINT "C(";I;")" ; C(I)
IIØ NEXT I
I2Ø END
```
} цикл замены нулевых элементов единицами и вывода массива C на печать

Вопросы для повторения

I. Что такое массив? Какие массивы могут использоваться в Бейсике?

2. Для чего служит оператор DIM ?

3. Можно ли описать несколько массивов в одном операторе DIM ? Приведите примеры.

4. Как образуются имена массивов?

5. Как обратиться к текущему элементу массива?

6. Каким образом вводятся элементы массива в память ЭВМ?

Упражнения

I. Запишите на Бейсике обращение к I5-му по порядку элементу целочисленного массива C.

2. Составьте программу вычисления среднего арифметического значения в одномерном массиве из I2 целых чисел.

3. Составьте программу переписи увеличенных в IØ раз значений элементов массива A в массив C. Исходный массив A содержит IØ чисел.

Полученный массив C распечатать на экране.

4. По приведенной программе сформулируйте условие задачи:

```
IØ DIM A$ (2ØØ)
2Ø FOR I = 1 TO  2ØØ
25 INPUT A$(I)
3Ø IF A$(I) = "D" OR  "Д" THEN  A$(I) = "␣"
4Ø NEXT I
```

5. Определите наименьший элемент в массиве, состоящем из N чисел.

6. Составьте программу, которая подсчитает, сколько раз встречается буквосочетание АБВ в тексте из N символов (N > 3).

IO. СИМВОЛЬНЫЕ ПЕРЕМЕННЫЕ

Наряду с числовыми переменными в Бейсик-АГАТЕ имеется возможность использовать переменные, значениями которых являются конечные последовательности символов – так называемые символьные или литерные переменные.

Использование символьных переменных дает возможность решать на ЭВМ различные задачи невычислительного характера.

Символьные переменные обозначаются так же, как и числовые (последовательностью букв и цифр, которая начинается с буквы), но с добавлением знака $ в конце. Например, A $ (читается: "A символьное"), P $ и т.д.

Длина символьной переменной не должна превышать 255 символов. Можно использовать и массивы символьных величин, которые описываются так же, как и числовые массивы. Например, оператор
IØ DIM A(I5), KL$(5,7) описывает числовой и символьный массивы.

Значением символьной переменной (простой или с индексом) является последовательность символов, которая заключается в кавычки. Например, "МИР", "МОСКВА".

Значения символьных переменных можно задавать:

I. Оператором присваивания. Например,
```
7Ø  S$ = "МИРУ-МИР"
```

2. При помощи оператора ввода INPUT . Например,
```
3Ø INPUT  A $
```

3. При помощи оператора ввода READ-DATA . При этом строка символов в операторе DATA заключается в кавычки. Например,
```
IØ  DATA "ИВАНОВ", "ПЕТРОВ", "СИДОРОВ"
5Ø  READ A$ , B$ , C$
```

Для символьных переменных определена операция сочленения. В результате её выполнения значения двух символьных переменных объединяются в одно. Если, например, A$ = "МИНСК" B$ = "ГОРОД-ГЕРОЙ", то в результате выполнения команды M$ = A$ + "-" + B$ символьной (литерной) переменной M$ будет присвоен текст МИНСК - ГОРОД-ГЕРОЙ .

10.1. Стандартные функции для символьных величин

1. LEN (A$) - вычисляет количество символов в значении символьной переменной. Например, если A$ = "Бейсик", то при помощи команды C = LEN(A$) переменной C будет присвоено значение 6.

2. LEFT$(A$,X) - формирует строку из первых X символов аргумента A$. Например, если A$ = "ПРОГРЕССИЯ", то в результате выполнения команды P$ = LEFT$(A$,8) переменной P$ будет присвоено значение ПРОГРЕСС .

3. RIGHT$(A$,X) - формирует строку из последних X символов аргумента A$. Например, если A$ = "СИМВОЛЬНЫЕ", то в результате выполнения команды Q$ = RIGHT$(A$,7) символьной переменной Q$ будет присвоено значение ВОЛЬНЫЕ .

4. MID$ (A$,X,У) - формирует строку из У символов, начиная с X-того символа. Например, если A$ = "КАЛЕНДАРЬ", то в результате выполнения команды R$ = MID$(A$,6,3) символьной переменной R$ будет присвоено значение ДАР .

10.2. Вывод на экран ВКУ текстовой (символьной) информации

При построении разных программ, особенно обучающих и игровых, возникает необходимость выводить текстовую (символьную) информацию на экран ВКУ, размещая её по заранее составленной схеме.

Экран дисплея содержит 32 строки, в каждой из которых можно разместить по 32 символа. Режим принято называть нормальным, когда на черном фоне экрана высвечиваются белые символы. После загрузки операционной системы всегда включен нормальный режим.

Режим называют инверсным, когда фон экрана белый, а символы черные. Оператор INVERSE включает инверсный режим. Если после работы в инверсном режиме необходимо вновь вернуться к нормальному, то это делается оператором NORMAL .

Для очистки экрана служит оператор HOME ; он также распространяет действие предыдущего оператора по всему экрану. Например:

```
1Ø INVERSE
2Ø HOME
```

очищает экран и закрашивает в белый цвет.

Текст можно выводить на экран в цветном исполнении, использовав оператор RIBBON = K, где K - код цвета:

Код цвета	Цвет	Код цвета	Цвет
0	черный	4	синий
1	красный	5	сиреневый
2	зеленый	6	голубой
3	желтый	7	белый

Оператор FLASH включает мерцающий режим вывода информации.

Пример 1.

Программа:

```
1Ø REM ПРИМЕР 1
15 INVERSE
2Ø RIBBON = 2
3Ø HOME
4Ø FLASH
5Ø RIBBON = 1
6Ø ? "ПЭВМ АГАТ ГОТОВА К РАБОТЕ"
```

Пояснения:

15 INVERSE, 2Ø RIBBON = 2, 3Ø HOME — очистка экрана и закрашивание зеленым цветом

4Ø FLASH — включение мерцающего режима

5Ø RIBBON = 1 — включение красного цвета

6Ø ? "ПЭВМ АГАТ ГОТОВА К РАБОТЕ" — выдача сообщения на экран дисплея

Элементы текста можно располагать в определенных местах экрана ВКУ. Для этого используются операторы табулирования.

Вертикальная табуляция - перемещение курсора на указанную строку, с которой произойдет печать следующего выводимого символа (текста или значения), - осуществляется оператором VTABУ, где У - номер строки, $1 \leqslant У \leqslant 32$.

Горизонтальная табуляция - перемещение курсора на указанную позицию строки - осуществляется оператором HTABX или оператором TAB(X), где X - номер позиции в строке, $1 \leqslant X \leqslant 32$.

Пример 2.

Программа:

```
1Ø REM ПРИМЕР 2
2Ø INVERSE
3Ø RIBBON = 6
4Ø HOME
5Ø FLASH
```

Пояснения:

2Ø INVERSE, 3Ø RIBBON = 6, 4Ø HOME — очистка экрана и закрашивание голубым цветом

5Ø FLASH — включение мерцающего режима

```
6Ø KIBBON = I          включение красного цвета
7Ø VTAB IO             установка курсора в строке IØ
8Ø ? TAB(I5) "ВНИМАНИЕ!"   установка курсора в позицию I5
                       и печать сообщения
```

В о п р о с ы д л я п о в т о р е н и я

1. Как описываются в программе символьные величины? Массивы символьных переменных?

2. Как задаются значения символьных переменных?

3. Какие стандартные функции существуют для символьных величин?

4. Какие возможности существуют для вывода на экран ВКУ текстовой (символьной) информации?

У п р а ж н е н и я

1. С помощью операции сочленения, используя стандартные функции для символьных величин, составьте из частей слова "ИНТЕГРАЛ" слова: а) "ГАНТЕЛИ", б) "РЕНТГЕН", в) "ТИГР", г) "АГЕНТ" .

2. Напишите алгоритм, проверяющий, совпадают ли первые три буквы одного слова с последними тремя буквами другого слова: а) "СОКОЛ" и "КУСОК", б) "СОКОЛ" и "КОЛОС" .

3. Даны две символьные переменные:

A $ = "АГРО"

B $ = "ЭКСПОНАТ"

Используя стандартные функции, из этих величин составить слово C $ = "АГАТ" .

4. Вывести на экране ВКУ в мерцающем режиме зеленым цветом надпись "ПРИВЕТ!", расположив её в I5-й строке, начиная с I2-й позиции.

5. Ввести текущую дату и выдать её красным цветом в центре экрана, предварительно очистив его от посторонней информации и закрасив зеленым цветом.

11. ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ ПЭВМ "АГАТ"

Информация на экран дисплея может выводится не только в виде символов, но и в графическом виде. Изображение формируется из отдельных точек (позиций). Для определения местоположения точки на экране необходимо знать её координаты. Плоскость экрана дисплея можно представить в виде декартовой системы координат с осью У, направленной вниз (рис.3).

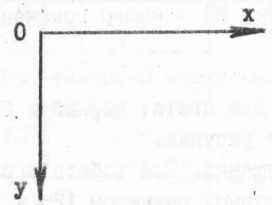


Рис. 3

При выводе графической информации ПЭВМ "АГАТ" может работать в трех режимах. Приведем таблицу, в которой даны графические режимы и приведены их основные характеристики.

Таблица I

Название режима	Цветность	Количество точек (позиций)	Способ включения
Графический высокого разрешения (ГВР)	черно-белый	256 x 256	HGR = N 1 ⩽ N ⩽ 7
Графический среднего разрешения (ГСР)	цветной	I28 x I28	MGR = N 1 ⩽ N ⩽ 7
Графический низкого разрешения (ГНР)	цветной	64 x 64	GR = N 2 ⩽ N ⩽ 3I

I. Графика высокого разрешения. При работе в режиме ГВР экран дисплея представляет собой матрицу размером 256 x 256 точек (рис.4).

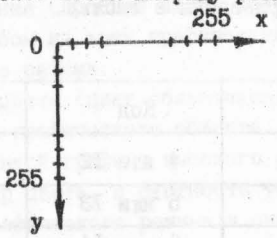


Рис. 4

Каждой точке экрана дисплея соответствует один бит информации оперативной памяти. Значит, экран отображает участок оперативной памяти объемом 256 x 256 бит (2^{16} = 65 536 бит). Этот участок памяти называется графической страницей. Таких участков выделяется несколько. Если значение бита оперативной памяти "0", точка на экране не светится, что соответствует черному цвету; если значение бита "I", точка светится белым цветом.

Для включения графического режима высокого разрешения используется оператор m HGR = N, где N - номер графической страницы ($1 \leqslant N \leqslant 7$).

Графику высокого разрешения иначе называют точной (тонкой) графикой. Она использует только два цвета: черный и белый. Черный - это цвет экрана, белый - цвет рисунка.

2. Графика среднего разрешения. При работе в режиме ГСР экран дисплея представляет собой матрицу размером I28 x I28 точек (рис.5).

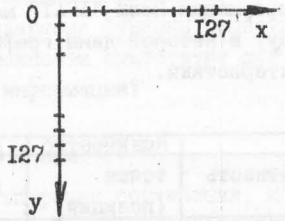


Рис. 5

Графический режим среднего разрешения включается оператором m MGR = N, где N - номер графической страницы ($1 \leqslant N \leqslant 7$).

Точка (позиция), с помощью которой получается изображение в этом режиме, в 4 раза больше по размеру, чем в точной графике (ГВР), и соответствует четырем битам информации оперативной памяти.

Преимуществом графического режима среднего разрешения является то, что он цветной. Для задания цвета используется оператор m COLOR = X, где X - код цвета.

Каждая точка цветной графики может быть окрашена в один из восьми цветов, коды которых приведены в табл.2.

Таблица 2

Код	Цвет	Код	Цвет
Ø или 8	черный	4 или I2	синий
1 или 9	красный	5 или I3	фиолетовый
2 или 1Ø	зеленый	6 или I4	голубой
3 или 11	желтый	7 или I5	белый

3. Графика низкого разрешения. При работе в режиме ГНР экран дисплея представляет собой матрицу размером 64 x 64 точек (рис.6).

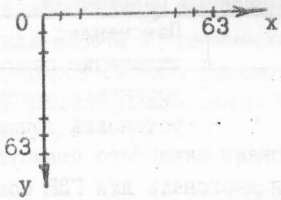


Рис. 6

Графический режим низкого разрешения включается оператором m GR = N, где N - номер графической страницы ($2 \leqslant N \leqslant 31$).

Точка в графике низкого разрешения в 4 раза больше точки ГСР. ГНР является цветной и иначе называется грубой графикой:

точка ГНР | точка ГСР | точка ГВР

Создание графического объекта производится оператором PLOT. Оператор PLOT имеет следующие модификации:

I. Установка точки текущего цвета с координатами X, У

m PLOT X, У

2. Вычерчивание отрезка текущего цвета от точки с координатами XI, УI к точке с координатами X2, У2

m PLOT X1, У1 TO X2, У2

Координатами точек могут быть константы, переменные или арифметические выражения (дробная часть координаты отбрасывается).

Работа в любом из трех графических режимов предусматривает:

I. Включение режима.

2. Задание цвета (цвет обязателен!).

3. Создание графического объекта.

Для черно-белой графики высокого разрешения COLOR = 1, причем 1 - это не номер цвета, а плотность точек на экране.

Выйти из графического режима в символьный можно двумя способами:

I. С помощью директивы TEXT = N, где N - номер текстовой страницы ($0 \leqslant N \leqslant 63$).

2. Одновременно нажать клавиши УПР и СБР.

Включение графической страницы автоматически производит очистку экрана дисплея.

Рассмотрим несколько примеров.

Пример I. Установить на экране точку с координатами X = 2Ø и

У = 2Ø в графическом режиме низкого разрешения.

```
Команды :                          Пояснения:
IØ GR = 2                          включение режима
2Ø COLOR = 2                       установка зеленого цвета
3Ø PLOT 2Ø,2Ø                      установка точки
```

Пример 2. Провести диагонали экранного прямоугольника в ГВР, ГСР, ГНР. Заметим, что левая диагональ для ГВР соединяет точки Ø,Ø и 255,255, а правая - точки с координатами Ø,255 и 255,Ø. Пусть диагональ будет зеленого цвета.

```
Программа 1                        Пояснения :
IØ HGR = 1                         включение ГВР
2Ø COLOR = 1                       установка плотности точек
3Ø PLOT Ø,Ø TO 255,255             построение отрезка
4Ø PLOT Ø,255 TO 255,Ø             построение отрезка
5Ø END                             конец
Программа 2
IØ MGR = 1                         включение ГСР
2Ø COLOR = 2                       включение зеленого цвета
3Ø PLOT Ø,Ø TO I27,I27
4Ø PLOT Ø,I27 TO I27,Ø
5Ø END
Программа 3
IØ GR = 2                          включение ГНР
2Ø COLOR = 2
3Ø PLOT Ø,Ø TO 63,63
4Ø PLOT Ø,63 TO 63,Ø
5Ø END
```

Пример 3. Начертить контуры параллелепипеда (рис.7).

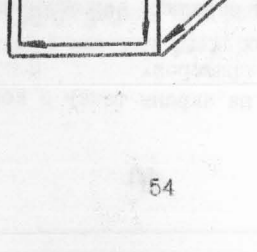


Рис. 7

Нарисуйте на бумаге контур параллелепипеда. Постройте координаты его вершин (для работы в графическом режиме среднего разрешения). Продумайте порядок обхода. Рассмотрите разные варианты обхода. Необходимо выбрать рациональный способ (с меньшим числом разрывов и без повтора обвода).

```
Программа:                         Пояснения :
IØ MGR = 1                         режим ГСР
2Ø COLOR = 2                       цвет зеленый
3Ø PLOT IØ5,I5 TO 75,45            построение ломаной от точки
  TO 75,IØ5 TO I5,IØ5              X = IØ5, У = I5 к точке
  TO I5,45 TO 75,45                X = 75, У = 45
4Ø PLOT I5,45 TO 45,I5 TO          построение ломаной от точки
  IØ5,I5 TO IØ5,75 TO 75,IØ5       X = I5, У = 45 к точке
                                   X = 75, У = IØ5
```

В данной программе можно предусмотреть выход из графического режима в текстовый, если ввести строку 5Ø TEXT = Ø . Следует заметить, что в данном случае изображение очень быстро исчезнет с экрана. Для задержки изображения можно использовать оператор GET.

Общий вид этого оператора:

m GET A$, где GET (дать) - служебное слово.

В результате работы этого оператора выполнение программы приостановится и продолжится только после ввода любого символа с клавиатуры.

Используем в программе строки

```
5Ø GET A$
6Ø TEXT = Ø
```

С помощью оператора GET запрашивается значение символьной переменной A$. После нажатия клавиши (введение любого символа) сработает оператор TEXT и произойдет переход в текстовый режим (изображение исчезнет с экрана).

Пример 4. Видоизменить программу примера 3 так, чтобы менялась окраска контура параллелепипеда (изображение переливается различными цветами).

```
Программа:                         Пояснения:
IØ MGR = 1
I5 FOR I = 1 TO 7                  начало оператора цикла
2Ø COLOR = I
3Ø PLOT IØ5,I5 TO 75,45 TO
  75,IØ5 TO I5,IØ5 TO
  I5,45 TO 75,45
```

```
4Ø PLOT  I5,45  TO  45,I5  TO
   IØ5,I5  TO  IØ5,75  TO  75,IØ5
5Ø NEXT I : GOTO I5                конец цикла, переход к строке I5
6Ø END                             конец
```

Программа зацикливается (для завершения необходимо сделать сброс). Для задержки изображения в очередном цвете необходимо заменить в программе строку 5Ø GET A% : NEXT I : GOTO I5 .

Пример 5. Закрасить экран в графике низкого разрешения.

```
Программа:                         Пояснения:
IØ HOME : VTAB I5 : PRINT "НОМЕР ЦВЕТА"  очистка экрана. Печать слов
                                   "Номер цвета"  начиная с
                                   позиции I5
2Ø GET X                           запрос значения цвета
3Ø GR = 2                          режим ГНР
4Ø COLOR = X                       установка цвета (по X)
5Ø FOR У = Ø  TO  63               цикл с параметром У
6Ø PLOT Ø,У TO  63,У               проведение горизонтальной
                                   прямой (внутри цикла)
7Ø NEXT У                          конец цикла
8Ø END                             конец программы
```

Пример 6. Провести горизонтальные линии на экране.

Рассмотрим режим ГВР и построение линий с шагом I0.

```
Программа:                         Пояснения
IØ HGR = 1                         режим ГВР
2Ø FOR I = Ø  TO  I5               цикл с параметром I (от Ø до I5)
3Ø COLOR = 1                       установка плотности точек
4Ø PLOT Ø,1Ø * I  TO  255,1Ø * I   горизонтальная линия
5Ø NEXT I : GET A o                конец цикла, запрос значения
6Ø END
```

Пример 7. Написать программу движения точки по горизонтали, пусть точка меняет цвет. Программу зациклить.

В программе будем использовать закрашивание следа точки в цвет экрана (черный).

```
Программа:                         Пояснения:
IØ REM ДВИЖУЩАЯСЯ ТОЧКА            комментарий
2Ø MGR = 1                         режим ГСР
3Ø FOR I = Ø  TO  7                начало оператора цикла
                                   с параметром  I
4Ø COLOR = I : GET У               номер цвета значения I,
                                   запрос значения У
```



```
5Ø FOR X = Ø  TO  I27              начало оператора цикла
                                   с параметром X
6Ø PLOT X, У                       установка точки
7Ø COLOR = Ø                       установка черного цвета
8Ø PLOT X, У                       установка точки
9Ø NEXT X                          окончание цикла с параметром X
IØØ NEXT I                         окончание цикла с параметром I
IIØ GOTO  3Ø                       переход к строке 3Ø
```

Пример 8. В режиме ГСР нарисовать сетку (решетку) с шагом в IØ позиций.

```
Программа:                         Пояснения:
IØ MGR = 1 : COLOR = 4             режим ГСР, цвет синий
2Ø FOR I = Ø  TO  I27              цикл с параметром I, изменяющимся
                                   от Ø до I27 с шагом IØ
3Ø PLOT Ø,I  TO  I27,I             горизонтальная линия
4Ø PLOT I,Ø  TO  I,I27             вертикальная линия
5Ø NEXT I                          конец цикла
6Ø GET A% : TEXT = Ø               запрос значения, включение
                                   текстовой страницы
```

Вопросы для повторения

I. Перечислите графические режимы, используемые при работе на ПЭВМ "АГАТ".

2. Каким образом выключаются известные Вам графические режимы?

3. Назначение оператора COLOR = X и является ли он обязательным при работе в любом графическом режиме.

4. Назначение оператора PLOT и его модификаций.

5. Какие способы выхода из графического режима в символьный Вы знаете?

Упражнения

I. Разделите экранный прямоугольник на 4 части с помощью горизонтальной и вертикальной прямых, соединяющих середины противоположных сторон экранного прямоугольника.

2. Написать программу, которая позволяет выполнить следующее:

а) квадрат зеленого цвета со стороной 60 разделить на 4 части и закрасить эти части в белый, желтый и синий цвета (контур квадрата должен быть сохранен);

б) осуществить движение точки по вертикали (цвет точки меняется);

в) нарисовать три квадрата разного цвета (контурно) так, чтобы каждый следующий квадрат был внутри другого;

г) используя упражнение 2,в , закрасить каждый квадрат, начиная с внешнего, в цвет контура;

д) нарисовать квадрат со стороной 120 и, разделив его по диагонали на две части, закрасить эти части красным и синим цветом.

12. КОМАНДА ОБРАЩЕНИЯ К ПОДПРОГРАММЕ

Язык Бейсик-АГАТ обладает возможностью повторения действия группы любых нужных операторов и возвращения в исходное место выполнения программы. Таким средством является обращение к подпрограмме.

Подпрограммой будем называть группу операторов программы, которая оформляется как отдельная программа (в так называемом стандартном виде).

Из основной программы можно обращаться к подпрограмме с помощью специального оператора: m GOSUB n , где m - номер текущей строки; GOSUB - служебное слово (происходит от сокращения английских слов, означающих в переводе "идти к подпрограмме"); n - номер строки начала соответствующей подпрограммы.

Например, оператор 40 GOSUB 200 означает обращение к подпрограмме, начинающейся строкой с номером 200.

Каждая подпрограмма должна заканчиваться оператором RETURN возврата в основную программу (т.е. в программу, из которой было обращение в данную подпрограмму).

Оператор RETURN осуществляет возврат в то место основной программы, из которого был переход к подпрограмме (к оператору, следующему за GOSUB).

Пример 1. Используя алгоритм поиска большего из двух чисел, написать программу поиска большего из трех чисел.

Алгоритм:

```
алг  БИД (вещ α, β, γ)
     арг α , β
     рез
нач
     если α > β
          то γ := α
          иначе γ := β
     всё
кон
```


Программа на языке Бейсик:

```
10 REM БИТ
20 INPUT A, B, C
30 GOSUB 100
40  A = C
50  B = Z
60 GOSUB 100
70 PRINT "БОЛЬШЕЕ ИЗ ТРЕХ ЧИСЕЛ"; Z
80 END
90 REM ПОДПРОГРАММА
100 IF  A >= B  THEN  Z = A
110  Z = B
120 RETURN
```

Вопросы для повторения

1. Что такое подпрограмма?

2. Как записывается в общем виде команда обращения к подпрограмме?

3. Привести пример использования оператора GOSUB .

Упражнения

1. Написать программу поиска большего из пяти чисел с использованием подпрограммы сравнения двух чисел.

2. Обеспечить красивую выдачу на экран предложения "Основы информатики и вычислительной техники". Каждая строчка текста должна быть окружена рамочкой из "ж". Для печати рамочки воспользоваться специальной подпрограммой.

3. Написать программу сортировки одномерного массива, используя в ней подпрограмму поиска минимального элемента.

Словарь служебных слов

№ п/п	Служебные слова	Произношение	Назначение
I.	NEW	НЬЮ	Очистка памяти
2.	HOME	ХОУМ	Очистка экрана
3.	LIST	ЛИСТ	Вывод на ВКУ исходного текста программы
4.	DEL	ДЭЛ	Уничтожение строк программы в ОП
5.	DELETE	ДЭЛЭТЭ	Уничтожение файлов на ГМД
6.	RUN	РАН	Запуск программы на выполнение
7.	LOAD	ЛОАД	Загрузка программы в ОП
8.	CATALOG	КАТАЛОГ	Чтение перечня программ, хранящихся на ГМД
9.	SAVE	СЭЙВ	Запись содержимого ОП на ГМД
I0.	REM	РЭМ	Комментарии (пояснения)
II.	LET	ЛЭТ	Присваивание
I2.	PRINT	ПРИНТ	Печать
I3.	INPUT	ИНПУТ	Ввод исходных данных
I4.	DATA	ДЭЙТА	Данные } Ввод исходных данных
I5.	READ	РИД	Читать } Ввод исходных данных
I6.	DIM	ДИМ	Размерность
I7.	IF	ИФ	Если } Условный переход
I8.	THEN	ЗЭН	Тогда } Условный переход
I9.	GOTO	ГОУТУ	Идти к ... (безусловный переход)
20.	END	ЭНД	Конец
2I.	OR	О	Или
22.	AND	ЭНД	И
23.	NOT	НОТ	Не
24.	FOR	ФО	Для } Заголовок цикла
25.	TO	ТУ	К } Заголовок цикла
26.	NEXT	НЭКСТ	Конец цикла
27.	STEP	СТЭП	Шаг
28.	LEN	ЛЭН	Вычисление количества символов в значении символьной переменной
29.	LEFT	ЛЭФТ	Выборка заданного количества первых символов текстового аргумента

№ п/п	Служебные слова	Произношение	Назначение
30.	RIGHT	РАЙТ	Выборка заданного количества последних символов текстового аргумента
3I.	MID	МИД	Формирование строки из заданного количества символов, находящихся внутри текстового аргумента
32.	INVERSE	ИНВЭРС	Включение инверсного текстового режима
33.	RIBBON	РИБОН	Включение цвета в текстовом режиме
34.	FLASH	ФЛЭШ	Включение мерцающего текстового режима
35.	NORMAL	НОРМАЛ	Включение нормального текстового режима
36.	VTAB У	ВТАБ ИГРЕК	Перемещение курсора на строку с номером У
37.	HTAB X	АШТАБ ИКС	Перемещение курсора на позицию X в строке У
38.	TAB(X)	ТАБ ИКС	Перемещение курсора на позицию X в строке оператора PRINT
39.	GOSUB	ГОУСАБ	Обращение к подпрограмме
40.	RETURN	РЭТОН	Возврат из подпрограммы
4I.	HGR	АШ ЖЭ ЭР	Включение графического режима высокого разрешения
42.	MGR	ЭМ ЖЭ ЭР	Включение графического режима среднего разрешения
43.	GR	ЖЭ ЭР	Включение графического режима низкого разрешения
44.	PLOT	ПЛОТ	Создание графического объекта
45.	TEXT	ТЭКСТ	Включение текстовой страницы
46.	GET	ГЭТ	Ввести символ
47.	COLOR	КОЛОР	Включение цвета в графическом режиме

Возможные ошибочные ситуации

№ п/п	Сообщение	Русский перевод
I.	SYNTAX ERROR	Синтаксическая ошибка
2.	ILLAGAL VALUE ERROR	Ложное значение, ошибка
3.	DIVIZION BY ZERO	Деление на ноль
4.	LONG STRING ERROR	Длинная строка, ошибка
5.	I/O ERROR	Ошибка ввода/вывода
6.	NO DATA ERROR	Мало данных, ошибка
7.	NO FOR ERROR	Нэкст без фор
8.	REENTER	Повторите ввод
9.	UNDEF STATEMENT ERROR	Используется неописанная переменная
I0.	TYPE ERROR NAME	Ошибочно указан тип (имени, величины, файла)

Л И Т Е Р А Т У Р А

I. Криницкий Н.П. Алгоритмы вокруг нас. - М.: Наука, 1977. - 223 с.

2. Кетков Б.Л. Программирование на Бейсике. - М.: Наука, 1968. - 157 с.

3. "Информатика и образование", 1986-1987.

4. У о р т. Бейсик. - М.: Мир, 1967. - 255 с.

С О Д Е Р Ж А Н И Е

ПЭВМ "АГАТ" в УПК и школе
(Учебное пособие для учащихся)

о

Редактор Н.А. Д а ш к е в и ч

о

Подписано в печать 11.02.88 г. АЗ 44787.
Формат бумаги 60x84 I/I6. Усл. печ. л. 3,72.
Уч.-изд. л. 4,2. Тираж 3000 экз. Зак. № 608.
Цена 5 к.

о

Ротапринт типографии БелИИЖТа,
246022, г. Гомель, ул. Кирова, 34.