**Программирование на QBASIC.**

# Содержание

[Содержание 1](#_Toc350758219)

[Алгоритм 2](#_Toc350758220)

[Введение в язык программирования Qbasic 4](#_Toc350758221)

[Среда программирования QBasic. 6](#_Toc350758222)

[Алфавит языка 7](#_Toc350758223)

[Выражения и операции 9](#_Toc350758224)

[Оператор присваивания 11](#_Toc350758225)

[Операторы ввода-вывода 12](#_Toc350758226)

[Операторы перехода 14](#_Toc350758227)

[Ветвление в алгоритмах и программах. Операторы условного перехода 16](#_Toc350758228)

[Циклы в алгоритмах и программах 18](#_Toc350758229)

[Строковые переменные 22](#_Toc350758230)

[Массивы 25](#_Toc350758231)

[Сортировка массива по возрастанию 35](#_Toc350758232)

[Подпрограммы 39](#_Toc350758233)

[Работа с файлами 40](#_Toc350758234)

[Комбинированные типы. 41](#_Toc350758235)

[Операторы графики 43](#_Toc350758236)

# Алгоритм

**Алгоритм** – это последовательность действий понятных исполнителю и приводящая к заданному результату.

Слово «алгоритм» произошло от латинского слова algorifmi. В IX веке математик Мухаммед ибн Муса ал-Хорезми впервые в истории математики ввёл общие правила решения квадратных уравнений. В Европе при переводе имя автора переделали в Алгоритми. Ссылки на рецепты решений из книги Мухаммеда европейские авторы начинали со слов: «Так говорил Алгоритми ...». С течением времени сами рецепты для решения математических задач стали называть алгоритмами.

Точное математическое определение понятия «алгоритм» было выработано лишь в тридцатых годах XX века. Почему же до этого времени математики довольствовались интуитивным понятием алгоритма? Это связано с тем, что обычно понятие алгоритма встречалось в связи с конкретным решением задачи. Об алгоритме говорили лишь тогда, когда предлагался способ решения какого-либо класса задач. В начале XX века в математике накопилось большое количество задач, которые не поддавались решению, несмотря на то, что над ними думали первоклассные ученые. Возникло подозрение, что для некоторых из этих задач вообще не существует разрешающего алгоритма. Утверждение о неразрешимости того или иного класса задач можно было вывести, только имея точное определение алгоритма, надо было знать, не существование чего требуется доказать. Понятие алгоритма является не только центральным понятием теории алгоритмов, не только одним из главных понятий математики вообще, но одним из главных понятий современной науки. Более того, сегодня, с наступлением эры информатики, алгоритмы становятся одним из важнейших факторов цивилизации.

Исполнитель алгоритма - это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом.

Исполнителя характеризуют:

* среда;
* элементарные действия;
* система команд;
* отказы.

***Среда (или обстановка)*** — это «место обитания» исполнителя.

***Система команд.*** Каждый исполнитель может выполнять команды только из некоторого строго заданного списка - системы команд исполнителя. Для каждой команды должны быть заданы условия применимости (в каких состояниях среды может быть выполнена команда) и описаны **результаты выполнения команды.**

После вызова команды исполнитель совершает соответствующее ***элементарное действие*.**

***Отказы*** исполнителя возникают, если команда вызывается при недопустимом для нее состоянии среды.

Обычно исполнитель ничего не знает о цели алгоритма. Он выполняет все полученные команды, не задавая вопросов «почему» и «зачем».

В информатике универсальным исполнителем алгоритмов является ***компьютер.***

Основные ***свойства алгоритмов*** следующие:

***Понятность*** для исполнителя — исполнитель алгоритма должен знать, как его выполнять.

***Дискретность*** (прерывность, раздельность) — алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов).

***Определенность*** — каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

***Результативность*** (или конечность) — алгоритм должен приводить к решению задачи за конечное число шагов.

***Массовость*** — алгоритм решения задачи разрабатывается в общем виде, т. е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется ***областью применимости алгоритма*.**

Существует несколько форм представления алгоритмов:

* словесная;
* табличная;
* графическая;
* программа.

***Словесный способ*** записи алгоритмов представляет собой описание последовательных этапов обработки данных. Алгоритм задается в произвольном изложении на естественном языке.

Словесный способ не имеет широкого распространения, так как такие описания:

* строго не формализуемы;
* страдают многословностью записей;
* допускают неоднозначность толкования отдельных предписаний.

***Программа –*** запись алгоритма в виде последовательности операторов-команд некоторого языка программирования компьютера***.***

***Графический способ*** представления алгоритмов является более компактным и наглядным по сравнению со словесным. При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

Графический способ подразделяется на:

* рисунок;
* граф-схемы;
* блок-схемы.

Примеры представления алгоритмов в виде ***рисунков*** вы можете увидеть на упаковках продуктов быстрого приготовления, в инструкциях по использованию бытовой техники и пр.

Можно представить алгоритм в виде схемы или ***графа*** — вторая, более строгая, формализованная форма

В виде ***графа*** удобно представлять алгоритмы решения логических задач, задач по комбинаторике и пр.

***Граф***— геометрический объект, состоящий из вершин и соединяющих вершины линий и дуг.

Наиболее распространенной формой представления алгоритма является ***блок-схема****.* Для отображения алгоритма в виде блок-схемы используется стандартный набор графических объектов (блоков).

Если алгоритм предназначен для исполнения техническим устройством, например станком с числовым программным управлением или компьютером, он представляется в виде ***программы****.*

**Блок-схема алгоритма** - наглядное графическое изображение структуры алгоритма. Она строится из блоков, соединенных стрелками. Стрелки изображают последовательность вычислений.

# Введение в язык программирования Qbasic

**Транслятор** (англ. *translator* — переводчик) — это программа-переводчик. Она преобразует программу, написанную на одном из языков высокого уровня, в программу, состоящую из машинных команд.

Транслятор обычно выполняет также диагностику ошибок, формирует словари идентификаторов, выдаёт для печати тексты программы и т.д.

Трансляторы реализуются в виде компиляторов или интерпретаторов. С точки зрения выполнения работы компилятор и интерпретатор существенно различаются.

**Компилятор** (англ. *compiler* — составитель, собиратель) читает всю программу *целиком*, делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется.

**Интерпретатор** (англ. *interpreter* — истолкователь, устный переводчик) переводит и выполняет программу *строка за строкой*.

**Компиля́тор:**

* Программа или техническое средство, выполняющее компиляцию.
* Машинная программа, используемая для компиляции.
* Транслятор, выполняющий преобразование программы, составленной на исходном языке, в объектный модуль.
* Программа, переводящая текст программы на языке высокого уровня в эквивалентную программу на машинном языке.
* Программа, предназначенная для трансляции высокоуровневого языка в абсолютный код или, иногда, в язык ассемблера. Входной информацией для компилятора (исходный код) является описание алгоритма или программа на проблемно-ориентированном языке, а на выходе компилятора — эквивалентное описание алгоритма на машинно-ориентированном языке (объектный код).

**Компиляция:**

* Трансляция программы на язык, близкий к машинному.
* Трансляция программы, составленной на исходном языке, в объектный модуль. Осуществляется компилятором.

**Компилировать** - проводить трансляцию машинной программы с проблемно-ориентированного языка на машинно-ориентированный язык.

Процесс компиляции, как правило, состоит из нескольких этапов: лексического, синтаксического и семантического анализов, генерации промежуточного кода, оптимизации и генерации результирующего машинного кода. Помимо этого, программа, как правило, зависит от сервисов, предоставляемых операционной системой и сторонними библиотеками (например, файловый ввод-вывод или графический интерфейс), и машинный код программы необходимо связать с этими сервисами. Связывание со статическими библиотеками выполняется редактором связей или компоновщиком (который может представлять из себя отдельную программу или быть частью компилятора), а с операционной системой и динамическими библиотеками связывание выполняется при начале исполнения программы загрузчиком.

**Достоинство:**

программа компилируется один раз и при каждом выполнении не требуется дополнительных преобразований. Соответственно, не требуется наличие компилятора на целевой машине, для которой компилируется программа.

**Недостаток:**

Отдельный этап компиляции замедляет написание и отладку и затрудняет исполнение небольших, несложных или разовых программ.

**Интерпретатор**

**Интерпрета́тор** (языка программирования):

* Программа или техническое средство, выполняющее интерпретацию.
* Вид транслятора, осуществляющего пооператорную (покомандную) обработку и выполнение исходной программы или запроса (в отличие от компилятора, транслирующего всю программу без её выполнения).
* Программа (иногда аппаратное средство), анализирующая команды или операторы программы и тут же выполняющая их.
* Языковый процессор, который построчно анализирует исходную программу и одновременно выполняет предписанные действия, а не формирует на машинном языке скомпилированную программу, которая выполняется впоследствии.

**Простой интерпретатор** анализирует и тут же выполняет (собственно интерпретация) программу покомандно (или построчно), по мере поступления её исходного кода на вход интерпретатора.

**Достоинство** такого подхода является мгновенная реакция.

**Недостаток** — такой интерпретатор обнаруживает ошибки в тексте программы только при попытке выполнения команды (или строки) с ошибкой.

После того, как программа откомпилирована, ни сама исходная программа, ни компилятор более не нужны. В то же время программа, обрабатываемая интерпретатором, должна заново *переводиться* на машинный язык при каждом очередном запуске программы.

*Откомпилированные* программы работают быстрее, но *интерпретируемые* проще исправлять и изменять.

Каждый конкретный язык ориентирован либо на компиляцию, либо на интерпретацию — в зависимости от того, для каких целей он создавался.

[**Бейсик**](http://book.kbsu.ru/theory/chapter6/1_6_11.html#BASIC) создавался как язык для начинающих программистов, для которых построчное выполнение программы имеет неоспоримые преимущества.

Иногда для одного языка имеется *и компилятор*, *и интерпретатор*. В этом случае для разработки и тестирования программы можно воспользоваться интерпретатором, а затем откомпилировать отлаженную программу, чтобы повысить скорость ее выполнения.

**Машинный код** (также употребляются термины **собственный код**, или **платформенно-ориентированный код**, или **родной код**, или **нативный код** — от англ. **native code**) — система команд конкретной вычислительной машины, которая интерпретируется непосредственно микропроцессором или микропрограммами данной вычислительной машины.

**Программа** — это просто длинный список инструкций, выполняемых процессором.

Прямой поток выполнения команд может быть изменён инструкцией перехода, которая переносит выполнение на инструкцию с заданным адресом. Инструкция перехода может быть условной, выполняющей переход только при соблюдении некоторого условия.

**Низкоуровневый язык программирования** (**язык программирования низкого уровня**) — язык программирования, близкий к программированию непосредственно в машинных кодах используемого реального или виртуального (например, Java, Microsoft .NET) процессора. Для обозначения машинных команд обычно применяется мнемоническое обозначение. Это позволяет запоминать команды не в виде последовательности двоичных нулей и единиц, а в виде осмысленных сокращений слов человеческого языка (обычно английских).

Иногда одно мнемоническое обозначение соответствует целой группе машинных команд, выполняющих одинаковое действие над разными ячейками памяти процессора.

Кроме машинных команд языки программирования низкого уровня могут предоставлять дополнительные возможности, такие как макроопределения (макросы).

Общеизвестный пример низкоуровнего языка — **язык ассемблера,** хотя правильнее говорить о группе языков ассемблера. Более того, для одного и того же процессора существует несколько видов языка ассемблера. Они совпадают в машинных командах, но различаются набором дополнительных функций (директив и макросов).

Язык QBASIC (Beginner's All-purpose Instruction Code) разработан Джоном Кимини и Томасом Куртцем в Дартмутском колледже, США, в середине 1960 г.

QBASIC занимает особое место среди всех языков высокого уровня. С самого начала он задумывался как универсальный язык для начинающих, и средства программирования на QBASIC до сих пор включаются в комплект поставки ПК, - как обязательный элемент технологии.

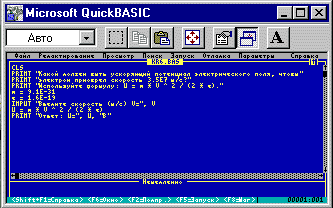
Вот некоторые достоинства QBASIC (с точки зрения массового пользователя):

* простота синтаксиса;
* простота организации данных и управляющих структур;
* большое число *встроенных* команд и функций, позволяющих без труда выполнять такие операции, как управление текстовым и графическим экраном, обработка символьных строк и т.п.)

Особым достоинством QBASIC следует считать возможность работы в режиме *интерпретации*, который резко упрощает процесс отладки программ: исполнение почти каждой команды можно проверить сразу после написания(Shift +F5).

# Среда программирования QBasic.

Бейсик (Basic) - диалоговый учебный язык программирования для персональных компьютеров. На современных компьютерах используется следующие версии Basic: GWBasic, QBasic 4.5, Turbo Basic для DOS, Visual Basic 3.0-6.0 для Windows. Здесь мы рассмотрим QBasic 4.5.



QBasic входит в состав операционной системы MS-DOS 6.0-6.22. Есть русифицированная версия QBasic.

Для запуска QBasic надо найти файл qbasic.exe и нажать Enter

Запуск программы Shift-F5.

Получение помощи Shift-F1.

Создать файл: Меню Файл, Сохранить Как, выбрать каталог, задать имя файла, ОК. Файл получит расширение bas.

Сохранить файл: Меню Файл, Сохранить.

Открыть файл, то есть считать его с диска: Меню Файл, Открыть.

Выход из QBasic: Меню Файл, Выход.

Копирование выделенного фрагмента программы в буфер (Edit,Copy) - Ctrl-Insert,

Вставка фрагмента из буфера (Edit,Paste) - Shift-Insert,

Вырезать (Edit,Cut) - Shift-Delete.

Удобно выделять и делать все указанные операции с помощью мыши. Можно выделенный фрагмент программы вставить в другой файл, если после выделения открыть его.

В QBasic 4.5 можно откомпилировать текстовый bas-файл, превратив его в двоичный exe-файл, который можно использовать отдельно, то есть без среды программирования QBasic.

# Алфавит языка

Алфавит языка Basic представляет собой таблицу символов ASCII. Первая половина этой таблицы (символы с кодом 0-127) - стандартная. Вторая половина (символы с кодом 128-255) специфична для каждой страны. В этой таблице каждый символ имеет 8-битовое обозначение. Итак, в алфавит языка Basic входят все прописные и заглавные буквы английского и русского алфавитов, цифры, а также набор специальных символов, который имеется на клавиатуре компьютера.

**Переменные.** В Бейсике различают переменные следующих типов:

* числовые переменные;
* символьные переменные;
* переменные пользовательского типа (записи);
* переменные-массивы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Тип переменной** | **Символ** | **Описание** | **Объём** | **Пример** |
| Целые числа | % | integer | 2 байта | 17; 123 |
| Действит. числа | нет | real | 4 байта | 3.1415 |
| Символьный | $ | string | min 1бай | язык |
| Пользовательский | нет | type |  |  |
| Целый 2 точности | # | double | 8 байт | 1.2543786 |

Переменная - это величина, которая может меняться при выполнении программы. Кроме указанных в таблице, целые числа записывают также в экспонецианальной записи, например,

**2,55.105=2.55Е5; 7,15.10-7=7.15Е-7**

Программа на языке Бейсик обрабатывает данные двух типов – числовые и символьные. Каждое число представлено в машине некоторой комбинацией битов. Любое число можно по-разному представить в машине:

1. целое число в диапазоне от –32768 до 32768;
2. длинное целое в диапазоне от –2147483648 до 21474833648;
3. вещественные числа обычной точности;
4. вещественные числа двойной точности;

Данные можно представить переменными и константами.

**Переменные.** С понятием переменной величины вы уже знакомы по урокам алгебры. Например, в простом алгебраическом равенстве ***c = f + 2b – 5*** значение *переменной с* зависит от значения переменной f и b, указанных в правой части равенства. Например, при ***f=2*** и ***b=6, с=9***.

Такое же равенство можно записать в программе на Бейсике.

***c = f + 2\*b – 5***

В терминах языка Бейсик ***c***, ***f*** и ***b*** – это имена переменных. Такие имена также называют *идентификаторами*.

**Идентификаторы**. В языке Бейсик идентификатор – это произвольный набор символов, который может содержать от 1 до 40 символов, причём первый символ должен быть латинской буквой, а остальные – латинские буквы или цифры или символы типа @, #, % и.т.д.

***Пример:***

***A, Ds, SodRan, k1, n123, dlina!***

Тип идентификатора в Бейсике опознаётся по последнему символу в имени переменной:

% - целое число;

& - длинное целое число;

! – вещественное число обычной точности;

# - вещественное число двойной точности;

$ - символьный тип.

В Бейсике предусмотрен другой способ описания типов переменных (qbasic). Если в начале программы поместить команду: DEFINT I – L, то все переменные, имена которых начинаются с буквы I и лежат в диапазоне до буквы L будут считаться целыми (INTEGER).

Общий формат команды описания типов:

1. DEFINT X-X (целые числа integer)
2. DEFLNG X-X (длинные целые числа long)
3. DEFSNG X-X (вещественные числа обычной точности, single)
4. DEFDBL X-X (вещественные числа двойной точности double)
5. DEFSTR X-X (символьный тип string)

**Константы.** Константы, как числовые так и символьные – это величины, которые не меняются в ходе выполнения программы.

***Например:***

***p= 3.1415***

***T$= “ПРИМЕР”***

***A=9575***

***m$=”Расписание на неделю”***

Иногда записанные константы называют *литералами.* В отношении констант необходимо помнить правила:

1. разделителем целой и дробной части является точка;
2. значения символьных констант заключается в кавычки.

Числа можно записать в *экспоненциальной* форме, например:

***0.0285=2.85Е-2*** или ***0.0285=2.85D-2***

***784.527=7.84Е+2*** или ***784.527=7.84D+2***

Числа, записанные в экспоненциальной форме, представляют собой произведение мантиссы на порядок, т. е. На 10 в степени –2, +2 и т. п. Буква Е используется для вещественных чисел обычной точности, буква D – для вещественных чисел двойной точности.

**Именованные константы.** Если присвоить переменной некоторое значение (числовое или символьное) и в дальнейшем не изменять эти значения, например:

***F3=60, n%=15***

тогда переменные F3 и n% можно считать именованной константой. В qbasic существует специальный оператор для описания именованной константы:

**CONST** <имя\_переменной = константа, имя переменной = константе>

Например: CONST F2=60, n%=15

# Выражения и операции

Последовательность операций, которые необходимо произвести над данными, чтобы получить требуемое значение, называется *выражением.* В Бейсике существует пять категорий операций:

1. *Арифметические операции;*
2. *Операции отношений;*
3. *Логические операции;*
4. *Функциональные операции;*
5. *Строковые операции.*

**Арифметические операции.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название операции** | **Знак**  **в матем.** | **Знак в Бейсике** | **Пример в**  **математ.** | **Пример в**  **Бейсике** |
| ***Ввозведение в степень*** | ***-*** | ***^*** | ***25*** | ***2^5*** |
| ***Сложение*** | ***+*** | ***+*** | ***5+10*** | ***5+10*** |
| ***Вычитание*** | ***-*** | ***-*** | ***a-b*** | ***a-b*** |
| ***Умножение*** | ***x;  .*** | ***\**** | ***2.5*** | ***2\*5*** |
| ***Деление*** | ***:*** | ***/*** | ***10:2*** | ***10/2*** |
| ***Целочисленное деление*** | ***:*** | ***\*** |  | ***10\3=3*** |

**Операции отношений.**

|  |  |  |
| --- | --- | --- |
| **Значение операции** | **Знак операции** | **Выражение в Бейсике** |
| ***Равенство*** | ***=*** | ***x=y*** |
| ***Неравенство*** | ***≠*** | ***x<>y*** |
| ***Меньше*** | ***<*** | ***x<y*** |
| ***Больше*** | ***>*** | ***x>y*** |
| ***Меньше или равно*** | ***≤*** | ***x<=y*** |
| ***Больше или равно*** | ***≥*** | ***x>=y*** |

**Функция** - это заранее определённая операция над данными. В Бейсике существует два вида функций: встроенные и определённые пользователем.

**Встроенные математические функции**

|  |  |  |  |
| --- | --- | --- | --- |
| **Название функции** | **Запись в**  **математике** | **Запись вБейсике** | **Пример в Бейсике** |
| **Абсолютная величина** | **|x|** | **ABS(X)** | **ABS(-5)** |
| **Экспонента** | **ex** | **EXP(X)** | **EXP(5)** |
| **Логарифм натуральн.** | **ln x** | **LOG(X)** | **LOG(3)** |
| **Остаток целочисленного деления (MOD)** | **-** | **x MOD y** | **19 MOD 6.7** |
| **Квадратный корень** | **skorodnoe_142** | **SQR(X)** | **SQR(4)** |
| **Синус** | **Sin x** | **SIN(X)** | **SIN(3.14)** |
| **Косинус** | **Cos x** | **COS(X)** | **COS(1)** |
| **Тангенс** | **tg x** | **TAN(X)** | **TAN(2)** |
| **Арктангенс** | **arctg x** | **ATN(X)** | **ATN(3)** |
| **Знак числового выражения** | **-** | **SGN(X)** | **SGN(-5)** |

Все остальные тригонометрические функции также как и гиперболические функции, определяются по известным математическим формулам, например, ***ARCSIN(X)=ATN(X/SQR(1-X^2))***.

При решении задач очень часто используются ещё две функции.

**1. RND(1)** - генератор случайных чисел, который задаёт случайные числа из интервала от 0 до 1.

***Пример:***

*Задайте случайное числа от 1 до 100. (Натуральное число).*

***N=(RND(1)\*100+1)***

**2. INT(числовое выражение)** - эта функция отбрасывает дробную часть при делении чисел.

***Пример:***

*В примере первом необходимо, чтобы числа N от 1 до 100 были натуральными.*

***N=INT(RND(1)\*100+1)***

**Некоторые основные директивы QBasic**

Директива REM или апостроф ' - обозначает комментарий к программе или команде. Вся строка (как правило, это поясняющий текст), что идёт после знака комментария, не исполняется. Этот знак используется при отладке программ.

Оператор CLS-оператор очистки экрана. С него часто начинаются программы, работающие в текстовом режиме.

Директива END означает конец программы. Применение этой команды в конце программы не является обязательным.

**Примеры**. Записать по правилам Бейсика математические выражения

1. (X^2+2\*X-5.12)/(X^2+12.51)

2) (COS(X)-SIN(X))/(ABS(COS(X)+SIN(X)))

**Упражнения**

**1. Какие из следующих последовательностей символов являются идентификаторами, а какие нет.**

а) Х б) Х1 в) Х′г) Х1Х2 д) AB е) ABCD

ж) SIN з) SIN ( Х ) и) А-1 к) 2А л) MAX15

**2. Записать по правилам Бейсика следующие выражения:**

а) б) в) ⏐ г)

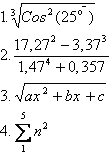
д) е) ж)

**3.** **Переписать следующие выражения, записанные по правилам Бейсика, в традиционной математической форме:**

а) А+B/ (C+D) - ( A+B) / C+D б) A\*B/ (C+D) - (C - D)/ B\*(A+B)

в) 1+SQR (COS (Х+Y)/ 2) г) 2.56 + АВS ( Х^2-Y^6)

д) INT (Х\*5.234 - А\*(Х+Y))



# Оператор присваивания

**Оператор присваивания имеет общий вид:**

**имя другой переменной**

**имя переменной**= **арифметические выражения**

**число**

В результате выполнения оператора присваивания переменной, стоящей слева от знака равенства присваивается значение другой переменной или значение арифметического выражения или численное значение. Для того, чтобы оператор присваивания мог быть выполним, необходимо, чтобы все переменные в выражении имели некоторые значения, были заданы выше.

**ПРИМЕРЫ**

1. А=0

2. С=2

3. А=2 : В=3

4. В=С

5. С= (А+В)^ 2/5

**УПРАЖНЕНИЯ**

**1. *Какие из следующих последовательностей символов являются операторами присваивания:***

а) А=В б) А\*Х+B=C в) Z=Z+1 д) Y=Y e) - Y=Y

**2. *Задать в виде оператора присваивания следующие действия:***

а) переменной Z присвоить значение, равное полусумме значений переменных Х и Y.

б) удвоить значение переменной А.

в) значение переменной Х увеличить на 0.1.

г) изменить знак значения переменной Y.

**3. *Задать с помощью операторов присваивания следующие действия:***

а) переменной А присвоить значение разности, а переменной В – полусуммы значений переменных Х и Y.

б) переменой А присвоить значение удвоенного произведения значений переменных Х и Y, а переменной В значение 0.

# Операторы ввода-вывода

**Оператор ввода с клавиатуры:**

**INPUT ["текст";] X1, [X2, X3... XN]**

[ ] - обозначают, что параметры, стоящие в них не обязательны;

**X1, X2, ... XN -** имена переменных или идентификаторы.

В результате выполнения этого оператора на экране появится текст, а переменным **X1, X2, ... XN** будут заданы значения с клавиатуры.

Пример: INPUT X - ввод числа X. На экране появится знак "?" и компьютер ждёт ввода переменной Х (это приглашение ввести число).

INPUT "Введите число Х="; X - можно вводить в кавычках подсказку, она будет выводиться на экран. После подсказки можно ставить знак "," вместо знака ";", но тогда знака вопроса "?" на экране не будет, а вместо него в качестве приглашения будет мигать курсор.

Пример: INPUT X - ввод числа X. На экране появится знак "?" и компьютер ждёт ввода переменной Х (это приглашение ввести число).

INPUT "Введите число Х="; X - можно вводить в кавычках подсказку, она будет выводиться на экран. После подсказки можно ставить знак "," вместо знака ";", но тогда знака вопроса "?" на экране не будет, а вместо него в качестве приглашения будет мигать курсор.

Пример: Вычислить площадь треугольника по формуле Герона.

CLS

'Определите площадь треугольника по формуле Герона.

INPUT "Введите величину стороны a: a=", a

INPUT "Введите величину стороны b: b=", b

INPUT "Введите величину стороны c: c=", c

p = (a + b + c) / 2

S = SQR(p \* (p - a) \* (p - b) \* (p - c))

PRINT "Ответ: площадь треугольника равна: S="; S

Пример (с использованием строковых переменных). Напечатайте на экране введённые с клавиатуры фамилию, имя и класс ученика.

CLS

INPUT "Введите свою фамилию: ", f$

INPUT "Введите своё имя: ", i$

INPUT "Введите свой класс: ", k$

PRINT

PRINT f$; " "; i$; ", "; k$

END

**Операторы блока данных**

**READ X1,X2,X3, ... XN** где X1,X2,..XN - это имена переменных

**DATA C1, C2, C3, ... CN** C1, C2, ... CN - это значения

В результате выполнения этих операторов переменная X1 получит значение C1, X2 примет значение C2...

*Замечание!* ОператорDATAможет стоять в любом месте программы. Считывать данные из хранилища DATA можно с помощью нескольких операторов READ.

Оператор **RESTORE** возвращает считывание данных на начало.

**Оператор вывода на экран**

**PRINT S1[ ;S2; S3 ...]**

где **S1, S2 ... SN** - списки

Списки это:

1. имя переменной, значение которой необходимо вывести на печать;
2. арифметическое выражение, значение которого необходимо вывести на печать;
3. текст, заключенный в кавычки.

Разделителем между списками может быть:

1. , - списки отделяются друг от друга шагом табуляции, равным 8 пробелам;
2. ; - списки печатаются вплотную друг к другу.

Для управления выводом используют специальный оператор:

LOCATE X, Y

где X - номер строки экрана 1<=X<= 24 , Y - номер столбца экрана 1<=Y<= 80.

Этот оператор указывает место для печати.

Просто оператор PRINT без текста и знаков даёт пустую строку на экране.

Если при работе в среде QBasic 4.5 вместо слова PRINT ставить знак вопроса "?", а затем нажать Enter, то слово PRINT само появится на экране. Это упрощает работу программиста.

Оператор LOCATE a, b - указывает позицию экрана (a-строка, b-столбец), начиная с которой будет печататься текст или число. Обычно используется перед оператором PRINT. Пример: LOCATE 4, 10: PRINT "Здравствуй, мир!"

**Задачи**

Написать программу для вычисления функции:

1) Y=,где Х=.

2) Y=, где А=- 5.25, В=, С=

3) Y=, где В=Sin, Х любое

1. Y где Х= (723,3/200,2)2 , Z=(53.2/41.3)3

5)Y= , где X=Sin2 |Z|+0.5Z ; A=2CosZ+1/(SinZ+2), где Z любое

6)Y=Cos, Z=Sin  ,X\*

7)Даны действительные числа Х,У. Не пользуясь никакими операциями, кроме умножения, сложения и вычитания, вычислить за минимальное количество операций:

а)3Х2У2-2ХУ2-7Х2У-4У2+15ХУ+2Х2-3Х+10У+6

б)1-2Х+3Х2-4Х3 в)1+2Х+3Х2+4Х3 г)2Х4-3Х2+4Х2-5Х+6

# Операторы перехода

Операторы условного перехода реализуют на языке программирования алгоритмическую конструкцию ветвления.

|  |  |  |  |
| --- | --- | --- | --- |
| **знаки сравнения** | | **логические операции** | |
| = равно | = | AND | логическое "и" (пересечение) |
| < меньше | < | OR | логическое "или" (объединение) |
| > больше | > | NOT | "не" (отрицание) |
| <=меньше или равно | <= |  |  |
| >=больше или равно | >= |  |  |
| <> не равно | <> |  |  |

Ветвление - это такая форма организации действий, при которой выполнение или иного действия зависит от выполнения или невыполнение некоторого условия.

Условие - это логическое выражение, содержащее знаки сравнения и логические операции (см. табл.)

Таблица действия логических функций (таблица истинности), где a и b -логические выражения.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | NOT a | NOT b | a AND b | a OR b |
| TRUE | TRUE | FALSE | FALSE | TRUE | TRUE |
| TRUE | FALSE | FALSE | TRUE | FALSE | TRUE |
| FALSE | TRUE | TRUE | FALSE | FALSE | TRUE |
| FALSE | FALSE | TRUE | TRUE | FALSE | FALSE |

Пример записи логического выражения: a + b > c AND d <> 8 .

Логическая операция NOT (отрицание) выполняется над простым логическим выражением:

NOT ( a> b )

Логическое выражение принимает одно из двух значений: TRUE (истина) и FALSE (ложь). Эти выражения применяются в операторах ветвления и цикла.

Существующий приоритет выполнения операций поддерживается и здесь:

сначала выполняются операции в скобках, затем логическое умножение, логическое сложение

Примеры записей логических выражений:

|  |  |
| --- | --- |
| математическая запись | запись на языке Бейсик |
| ( X>0 ) и (Y<0 ) | ( X>0 ) AND (Y<0 ) |
| -2<= X <=2 | (X>=-2 ) AND (Y<= 2 ) |
| X<0 и X>10 | ( X<0 ) OR (X>10 ) |

Принят следующий приоритет логических и арифметических операций:

* операции в скобках;
* операция NOT ;
* операции AND, /, \* ;
* операции OR, +, - ;
* операции отношения.

***Упражнения***

1. Вычислить значение логического выражения

а) Х2 -У2  <=12 при Х=0 и У=-1 б) NOT ((X\*Y) OR (Y >X) при X=4, Y=1

1. Написать логические выражения, которые имеют значение истинна лишь при выполнении указанных условий:

а) X>0 и Y>0 Y не является MAX (X, Y, Z)

б) неверно, что X>0 X, Y, Z равны между собой

в) 2< X < 20 точка с координатами X, Y принадлежит 1- ой четверти

Операторы перехода разделяются на две группы:

операторы безусловного перехода и операторы условного перехода.

**Оператор безусловного перехода.**

Он служит для перехода из одной строки программы к другой, помеченной номером или меткой.

Общий вид оператора:  **GOTO N**

где N - номер строки или метки оператора, на который происходит переход в программе.

**Оператор безусловного перехода GOTO N, где N-метка. В качестве N можно взять любое число, не обязательно совпадающее с номером строки, так как в QBasic 4.5 строки можно не нумеровать. Но двух одинаковых меток в одной программе не должно быть. Исполнение программы будет продолжено с той её части, которая начинается с указанной метки.**

**Примеры:**

1. X=3 2. GOTO W

GOTO 7 . . .

Y=2\*X W: PRINT "решение получено"

7: Z=5\*X

# Ветвление в алгоритмах и программах. Операторы условного перехода

Существуют две формы ветвления: полная и неполная.

нет

да

да

нет

Блок операторов

Блок операторов 1

Блок операторов 2

неполная форма ветвления

Полная форма ветвления

Соответственно существуют и два вида операторов: полной и неполной альтернативы.

* 1. **Полная альтернатива** (предусмотрены команды в ветви "ДА" и в ветви "НЕТ").

1. *Линейная форма записи* (оператор записывается в одной строке)

**IF условие THEN блок операторов 1 ELSE блок операторов 2**

Работа оператора: в зависимости от того выполняется или не выполняется условие (*условие - логическое выражение, которое может содержать знаки сравнения и логические операции)*. Если условие истинно, то выполняется блок операторов 1, если условие ложно, то выполняется блок операторов 2.

**ПРИМЕР:** IF A=1 THEN PRINT "ДА" ELSE PRINT "НЕТ"

PRINT "КОНЕЦ"

Здесь операторы работают следующим образом:

1. если условие А=1 "ИСТИННО", то выполняются операторы PRINT "ДА" и PRINT "КОНЕЦ";
2. если условие А=1 "ЛОЖНО", то выполняются операторы PRINT"НЕТ" и PRINT "КОНЕЦ".
3. *Блочная форма записи:*

**IF условие THEN** блок операторов ветви "ДА" **ELSE** блок операторов ветви "НЕТ"

* 1. **Условный оператор неполной альтернативы:**

**IF условие THEN блок операторов**

Если заданное условие истинно, выполняется список операторов, стоящий после THEN, в противном случае выполняется список операторов, стоящий после ELSE.

Слово ELSE может и отсутствовать. Тогда происходит переход к строке, следующей за оператором IF...THEN.

Условия представляют собой логические соотношения: равенства или неравенства = равно, <> неравно, > больше, < меньше, >= больше или равно, <= меньше или равно.

При использовании сложных условий (объединении двух и более условий) применяют слова AND (И) и OR (ИЛИ): Условие1 AND (или OR) Условие2...

Условный оператор позволяет реализовать алгоритм с ветвлением.

**Примеры решения задач.**

***Пример 1***.Определить наибольшее из двух чисел Х и У, введенных клавиатуры.

CLS: INPUT " ";X, Y

IF X > Y THEN

PRINT "НАИБОЛЬШЕЕ ЧИСЛО X="; X

ELSE

PRINT "НАИБОЛЬШЕЕ ЧИСЛОY=";Y

END IF

END

***Пример 2****.* Вычислите Функцию для любого значения Х.

X2 +4X-7 , X<=-2 CLS: INPUT "введите X";X

Y=< IF X<=-2 THEN

X2-3X+12 , X>2 Y=X^2+4\*X-7

ELSE

Y=X^2-3\*X+12

END IF

PRINT "ФУНКЦИЯ =";У;" при Х=";Х

END

***Пример 3****.* Вычислите функцию У=3Х2 -5Х+12 на отрезке (-3,3) с шагом 1.

CLS: PRINT "Y", "X"

X=-3

4 Y=3\*X^2-5\*X+12

PRINT Y,X

X=X+1

IF X<3 THEN GOTO 4

END

***Пример 4.*** Вычислите сумму целых чисел от 10 до 20

S =10+11+12+...+20

CLS: S=0: X=10

5 S=S+X

X=X+1

IF X<=20 THEN GOTO 5

PRINT "сумма = ";S

END

***Пример 5.*** Вычислите произведение нечетных чисел от 15 до 30

Р= 15\*17\*19\*...\*29

CLS:P=1:X=15

7 P=P\*X

X=X+2

IF X<30 THEN GOTO 7

PRINT "произведение = "; P

END

# Циклы в алгоритмах и программах

В языке QBASIC предусмотрено два основных способа организации циклов:

* повторение блока команд заданное количество (число) раз (цикл со счетчиком);
* циклическое повторение блока команд, пока выполняется (или не выполняется) некоторое условие.

**Цикл со счетчиком (FOR... NEXT) ДЛЯ... СЛЕДУЮЩИЙ**

Оператор цикла реализует алгоритмическую конструкцию, в которой определенные действия повторяются многократно.

Общий вид оператора цикла со счетчиком:

FOR K=K1 TO K2 STEP K3 - заголовок цикла

- тело цикла -

NEXT K - конец цикла

**То, что стоит между началом и концом цикла, называется телом цикла. Иногда также применяются вложенные циклы, то есть циклы могут быть вложенными один в другой.**

Здесь FOR - имя оператора (переводится "для"); К - переменная (обычно целого типа), называемая *переменная управления циклом* ; К1, К2, К3 - начальное, конечное значения и шаг изменения величины К соответственно, могут быть представлены любыми арифметическими выражениями; ТО, SТЕР - переводится - "до", "шаг"; тело цикла - любые операторы языка Бейсик; NEXT - оператор (переводится "следующий").

Совместная работа операторов и определяется схемой:

K=K1

K<K2

Тело цикла

К=К+К3

Да

Нет

Иначе говоря, операторы FOR и NEXT обеспечивают изменение значения К от К1 до К2 с шагом К3 и выполнение оператора (операторов), заключенных между FOR и NEXT, при каждом значении К.

***Ограничения:***

1. Изменять значение величин К1, К2, К3 в процессе выполнения операторов FOR и NEXT не рекомендуется.
2. Вход в цикл, минуя оператор FOR, в Бейсике запрещен.

**Примеры решения задач**

***Пример 1*.** Выведите на экран в столбик целые числа от 1 до 10.

CLS

FOR J=1 TO 10 STEP 1

PRINT J

NEXT J

END

***Пример 2*.** Вычислите сумму целых чисел от 1 до 10

CLS: S=0

FOR I=1 TO 10 STEP 1

S=S+I

NEXT I

PRINT " "; S

END

***Пример 3.*** Вычислите значение функции y=2X2-3X+5 на отрезке (-3, 3) с шагом 1,5

CLS : PRINT "Y"; "X"

FOR X=-3 TO 3 STEP 1.5

Y=2\*X^2-3\*X+5

PRINT Y,X

NEXT X

END

***Пример 4.*** Вычислите сумму S= 2+4+ ...+2 N , если N вводится с клавиатуры CLS : S=0

INPUT «введите « N»; N

FOR I = 1 TO N

S=S + 2 \*N

NEXT I

PRINT «сумма = «; S

END

***Пример 5***  Выведите слово «IBM» 10 раз, чтобы получилась диагональ

CLS

FOR K =5 TO 15

LOCATE K, K

PRINT « IBM»

NEXT K

END

***Пример 6.*** Напечатать на экране текст 20 раз.

CLS

FOR X=1 TO 20

PRINT "С Новым Годом!"

NEXT X

***Пример 7.*** Вычислить сумму n первых членов ряда S=1+1/4+1/9+...+1/n^2

CLS

INPUT "Введите число членов ряда: n=", n

FOR i = 1 TO n

S = S + 1 / i ^ 2

NEXT i

PRINT "Сумма n первых членов ряда равна: S=", S

***Пример 8*.** Нарисуйте 15 разноцветных концентрических окружностей.

SCREEN 7

FOR i = 1 TO 15

CIRCLE (160, 100), i \* 6, i

NEXT i

***Пример 10.*** Разлинуйте экран в клеточку.

SCREEN 7

FOR i = 1 TO 32

LINE (0, 10 \* i)-(320, 10 \* i), 15

LINE (10 \* i, 0)-(10 \* i, 200), 15

NEXT i

Одновременное использование циклического и условного операторов. Вычисление функций, имеющих ограничения в области определения.

Примечание: Оператор IF...THEN...ELSE в QBasic должен быть записан в одну строку!

***Пример 11.***

CLS

REM Вычислить функцию y=1/(x^2-5\*x-6)

FOR x = -10 TO 10

IF x ^ 2 - 5 \* x - 6 = 0 THEN PRINT "При x=-1, x=6 функция не определена!"

ELSE y = 1 / (x ^ 2 - 5 \* x - 6)

PRINT "x="; x, " y="; y

NEXT x

***Пример 12.***

CLS

REM Вычислить функцию y=1/SQR(x^2-3\*x+2)

FOR x = -5 TO 5

IF x >= 1 AND x <= 2 THEN PRINT "При 1<=x=<2 функция не определена!"

ELSE y = 1 / SQR(x ^ 2 - 3 \* x + 2)

PRINT "x="; x, " y="; y

**Цикл WHILE... WEND (ПОКА... КОНЕЦ)**

Конструкция WHILE... WEND (ПОКА... КОНЕЦ) позволяет выполнять блок команд до тех пор, пока значение выражения, заданного в WHILE - "Истина".

*Общий вид оператора*

**WHILE логическое выражение**

**-- блок команд –**

**WEND**

Работа операторов может определяться следующей схемой:

Тело цикла

Условие

Да

Нет

После завершения цикла управление передается команде, следующей за WEND.

Если при проверке условия окажется, что значение логического выражения - "Ложь", блок команд не будет выполнен *ни разу*.

С помощью WHILE... WEND можно организовать вложенные циклы - каждый со своими WHILE... WEND.

***Ограничение*:** из этой конструкции нельзя выйти "досрочно", т.е. до завершения работы оператора.

**Примеры решения задач.**

CLS : I=1

WHILE I < 5

PRINT "I=" ; I;

I=I+1

WEND

На экране будет : I=1 I=2 I=3 I=4

**Задачи по теме "Условный оператор" и "Оператор цикла**"

***Пример 1***. Найти наибольшее из чисел (х+у)/2 и (х+у/2)для любых чисел х и у.

***Пример 2.*** Вычислить значения функций для любых Х и Z

а)Y= (X-15)/(x-6) ;

б) Y= 2X-5/(7Z-4X);

с) Y=

г) Y=

д) Y=

***Пример 3.*** Заданы длины сторон треугольника - А, В, С, Определить, является ли треугольник равнобедренным.

***Пример 4.*** Составить программу, выводящую на экран первые десять натуральных чисел.

***Пример 5.*** Составить программу, печатающую таблицу квадратов первых десяти целых чисел в центре чистого экрана.

***Пример 6.*** Вычислить значения функции

а) Y= 5\*X2 -4X+11 на отрезке (-5,5) с шагом 1,5;

б) Y= на отрезке (-7, 7) с шагом 1;

в) Y= на отрезке (-6, 6) с шагом 0,5.

***Пример 7.*** Выведите в столбик ряд натуральных чисел от 1 до 10.

***Пример 8.*** Выведите на экран в столбик четные числа от 1 до 20.

***Пример 9.*** Выведите на экран в строку нечетные числа от 15 до 20.

***Пример 10.*** Найдите:

1. сумму целых чисел от 10 до 20;
2. произведение целых чисел от 20 до 20;
3. сумму нечетных чисел от 20 до 40;
4. произведение целых чисел от 20 до 40;
5. сумму четных чисел от 1 до 20;
6. произведение четных чисел от 1 до 20.

***Пример 11.*** Выведите на экран 10 раз слово "PENTIUM" по диагонали.

***Пример 12.*** Дана последовательность чисел, введенная с клавиатуры. Найдите сумму отрицательных членов этой последовательности и произведение положительных, количество отрицательных и положительных.

***Пример 13.*** Составьте программу, выводящую на экран:

1. движение "\*" по первой строке экрана;
2. движение "\*" по последнему столбцу экрана;
3. движение "\*" по диагонали.

***Пример 14.*** Вычислите суммы, если N вводится с клавиатуры:

1. ++…+
2. ++…+

***Пример 15.*** Вычислите произведения, если a и n вводятся с клавиатуры:

1. ++…+
2. a\*(a-n)\*(a-2n)...(a-n\*n);
3. 2\*4\*6\*...\*(2\*n) .

***Пример 16.*** Определить является ли произвольное число кратным:

а) двум; б) трем; в) пяти.

***Пример 17.*** Выведите на экран простые числа в диапазоне

а) от 1 до 20; б) от 20 до 100.

***Пример 18.*** Определить является произвольное число простым.

# Строковые переменные

В языке Бейсик существует возможность обработки не только числовых, но текстовых, символьных данных. Для этого используют символьные (или строковые, литерные) константы, переменные и выражения.

Символьная константа - это строка символов, заключенная в кавычки. Значением константы является последовательность составляющих ее символов не считая кавычек. Два идущих подряд знака кавычек задают пустую символьную строку, не содержащую ни одного символа.

У символьной переменной есть имя, которое заканчивается знаком $, и значение в виде строки символов.

Для строк определена операция сравнения: строки считаются равными, если их длины равны и коды всех символов попарно совпадают. Если одна из строк совпадает с началом другой (но короче ее), то она меньше. В остальных случаях все решает код первого несовпадающего символа - меньше та строка, у которой он меньше.

**Операции, производимые над строковыми переменными**

а) присвоение текстовой константы:

A$="СТРОКА СИМВОЛОВ"

T$="" - очистка строки

б) печать значений строковых переменных:

PRINT A$, T$

в) склейка двух строк:

T$=A$+B$

P$="СЛОВО: "+P$

C$="СКЛЕЙКА"+" "+"СТРОК"

г) ввод значений строковых переменных с клавиатуры:

INPUT A$, B$

- при вводе строку можно не заключать в кавычки.

**Основные функции работы с символьными переменными:**

**VAL(A$)** - перевод строки в число;

**CHR$(L)** - выдает символ по коду данного числа;

**ASC(S$)** - выдает код первого символа строки

**SRT$(K)** - преобразует число в символьную строку, которая является записью в виде последовательности десятичных цифр;

**LEN(F$)** - выдает длину символьного выражения;

**LEFT$( F$, N)** - вырезает заданное количество символов слева;

**RIGHT$ (F$, N)** - вырезает заданное количество символов справа;

**MID$(F$, N1, N2)** - вырезка из строки заданного количества символов с заданного символа

**Примеры решения задач.**

***Пример 1*.** Подсчитать, сколько раз среди символов заданной строки встречается буква "F".

CLS

INPUT "Введите строку - ";S$

K=0

FOR I=1 TO LEN(S$)

IF MID$(S$, I, 1)="F" THEN K=K+1

NEXT I

PRINT "Буква F встретилась ";K;" раз(а) "

END

***Пример 2*.** Составить программу, которая печатает заданное слово задом наперед.

CLS

INPUT "Введите слово - ";S$

FOR I=LEN(S$) TO 1 STEP -1

PRINT MID$(S$, I, 1);

NEXT I

END

***Пример 3.*** Выдать номер позиции заданной строки, в которой впервые встречается буква "в". Если такой буквы в строке нет, выдать сообщение об этом.

CLS: INPUT "Введите строку - ";S$

K=0

FOR I=1 TO LEN(S$)

IF MID$(S$, I, 1)="в" THEN 80

NEXT I

PRINT "Буквы в нет в заданной строке"

GOTO 90

80 PRINT "Первая буква в встретилась в "; I ; " -й позиции"

90 END

**Задачи на работу с символьными переменными**

1. Подсчитать сколько раз среди символов заданной строки встречается буква «F».
2. Составить программу, которая печатает заданное слово задом наперед.
3. Подсчитать количество цифр в заданной строке.
4. Подсчитать количество слогов в слове.
5. Проверить, является ли заданное слово палиндромом («перевертышем», как слова КАЗАК, ШАЛАШ и т.д.)
6. Среди натуральных чисел от 1 до n найти все такие, запись которых совпадает с последними цифрами записи их квадратов (5 и 25, 6 и 36, 25 и 625 и т.д.)
7. Выдать номер позиции заданной строки, в которой впервые встречается буква «в». Если такой буквы в строке нет, выдать сообщение об этом.
8. Выдать номер позиции заданной строки, в которой в пятый раз встречается буква «W». Если такой позиции нет, выдать сообщение об этом.
9. В заданной строке подсчитать отдельно количество символов «а» и «b».
10. Из заданной строки получить новую, заменив все запятые на точки. Распечатать полученную строку.
11. Из заданной строки получить новую, выбросив из нее все точки. Распечатать полученную строку.
12. Из заданной строки получить новую, удалив все звездочки («\*») и повторив каждый символ, отличный от звездочки. Распечатать полученную строку.
13. Из заданной строки получить новую, заменив в ней все символы после первой звездочки («\*») знаком «-» (если в строке нет звездочек, оставить ее без изменения). Распечатать полученную строку.
14. Из заданной строки получить новую, удалив из нее все двоеточия, предшествующие первой точке, и заменив в ней все знаки «+» после первой точки знаком « - » (если в строке нет точек, добавьте одну в конце). Распечатать полученную строку.
15. В заданной строке поменять местами первую букву « а » и первую букву « b ». Если хотя бы одной из них нет, напечатать сообщение, иначе распечатать полученную строку.
16. Определить наименьшую длину слова в заданной строке (считать, что слова отделяются друг от друга пробелами).
17. Из заданной строки выделить подстроку длиной 5 знаков (либо до конца строки, если осталось меньше), которая начинается с первой встретившейся в строке буквы ″ f ». Распечатать эту подстроку или сообщение о том, что ее нет.
18. В заданной строке первый символ - заглавная латинская буква. Если это « К », « О », «D», заменить ее соответствующей строчной буквой. Распечатать полученную строку.
19. В заданной строке подсчитать количество букв латинского алфавита.
20. Из заданной строки получить новую, удалив все символы, не являющиеся буквами латинского алфавита.
21. В заданной строке преобразовать все заглавные буквы латинского алфавита в строчные, а строчные в заглавные.
22. Из заданной строки получить новую, удалив все группы букв «abcd».
23. Из заданной строки получить новую, заметив все группы букв «abc» на «abcd».
24. Подсчитать, сколько слов в заданной строке оканчивается буквой «я», если известно, что после слова обязательно стоит пробел, либо точка.
25. Даны натуральное число n, символы s1,...sn. Преобразовать последовательность s1,...sn, , заменив в ней:

а) все восклицательные знаки точками;

б) каждую точку многоточием (т.е. тремя точками);

в) каждую из групп стоящих рядом точек одной точкой;

г) каждую из групп стоящих рядом точек многоточием (т.е. тремя точками).

1. Даны натуральное число n, символы s1,...sn. Выяснить, имеются ли в последовательности s1,...sn такие члены последовательности sj,sj+1, что sj - это запятая, а sj+1 - тире.
2. Даны натуральное число n, символы s1,...sn. Получить первое натуральное j, для которого каждый из символов sj и sj+1 совпадает с буквой а. Если такой пары символов в последовательности s1,...sn  нет, то ответом должно быть число 0.

# Массивы

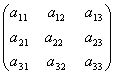
**Массив – это совокупность однотипных элементов**.

Математическими прообразами массивов являются векторы и матрицы с числовыми компонентами. В языках программирования все элементы массива объединяются общим именем – идентификатором массива. Элементы массива последовательно располагаются в памяти компьютера. При обращении к элементу массива указывается имя массива (идентификатор), индекс и тип. Если каждый элемент массива имеет только один индекс, то такой массив называют *линейным* или *одномерным*. Массивы, элементы которых имеют два индекса, называют *двумерными*. Например, массив А(8)

**А1, А2, А3, А4, А5, А6, А7, А8**

Этот массив является одномерным, причем массив имеет имя А и состоит из восьми элементов. Количество индексов в списке определяет размерность массива.

Двухмерный массив удобно представить в виде матрицы с двойными индексами.



В таком массиве каждый элемент имеет двойной индекс. Первая цифра индекса указывает номер строки, вторая номер столбца. Элементы, где номер строки и номер столбца одинаковы, называют элементами главной диагонали. Так как каждый элемент массива имеет индекс, то индекс показывает место элемента в массиве.

В некоторых языках программирования наряду с понятием массив используется и другое понятие *множество*. Отличие массива от множества в том, что массивы – это упорядоченные совокупности элементов, множества же представляют собой неупорядоченную совокупность отличных друг от друга элементов. Применительно к множествам допустимы операции добавления элемента к множеству, исключения элемента из множества, а также проверка того, принадлежит ли данный элемент рассматриваемому множеству.

Если двумерный массив имеет число строк равное числу столбцов, то такой массив (матрица) называют квадратным, например, массивы 3х3, 4х4, 5х5 и.т.д являются квадратными в отличие от массивов 2х4, 5х3, и.т.д.

В Basic для описания массивов применяют специальный оператор **DIM** (сокращение английского слова **DIMENSION** – размерность).

Оператор **DIM** используется для того, чтобы указать общее количество переменных в массиве. Это необходимо для рационального распределения памяти компьютера.

**DIM A(7)** количество переменных в массиве

Исполнение оператора DIM заключается в том, что в памяти компьютера образуется массив переменных.

А(0) А(1) А(2) А(3) А(4) А(5) А(6) А(7)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| А |  |  |  |  |  |  |  |

**Ограничение:** массив нельзя объявлять дважды, поэтому следует объявлять массивы в начальных строках программы и не возвращаться в эти строки с помощью оператора GOTO.

Каждый элемент массива обладает:

1. именем Х(5)
2. индексом
3. значением 3,2

Кроме оператора DIM для описания массива используется идентификатор (имя массива), тип массива и его размер. Например,

DIM A(7)

DIM V%(12)

DIM WR$(5)

В первом примере имя массива А, его тип – все действительные числа, размер – семь элементов. Во втором примере имя – V, тип массива – целые числа, их количество (размер массива) – 12 элементов; в третьем примере массив с именем WR является символьным. Этот массив содержит пять любых символов, например, букв, слов, и.т.п. Во всех трех примерах массивы были одномерными. Для двумерных массивов:

DIM AS(3,3)

DIM B%(2,5)

DIM c%(3,6)

имя и тип определяется аналогично одномерным, разница только в размерности. Например, массив AS имеет три строки и три столбца, то есть является квадратным. Если массив имеет менее десяти элементов, то во многих версиях Бейсика он задается по умолчанию. В момент описания массива (при обработке оператора DIM и выделению массиву места в памяти) значения элементов массива не определены; это значит, что в числовом массиве могут быть любые числа. Элементы массива могут быть введены с клавиатуры с помощью оператора INPUT или с помощью операторов DATA, READ или иным способом. Например,

10 DIM A(12)

20 FOR I=1 TO 12

30 INPUT A(I)

40 NEXT I

или

10 DATA 6,3,12,78,5

20 DIM A(5)

30 FOR I=1 TO 5

40 READ A(I)

50 NEXT I

Вывод на печать элементов массива также осуществляется с помощью цикла с параметром.

**Пример**

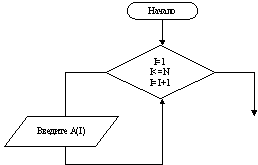
Вывести на печать элементы массива

60 FOR I=1 TO 5

70 PRINT A(I)

80 NEXT I

Результатом работы этой программы будет пять чисел, которые будут выведены на экран в столбик. Если в конце строки 70 поставить “;” тогда элементы массива будут напечатаны на экране в строку. С помощью блок-схемы алгоритм ввода можно представить.

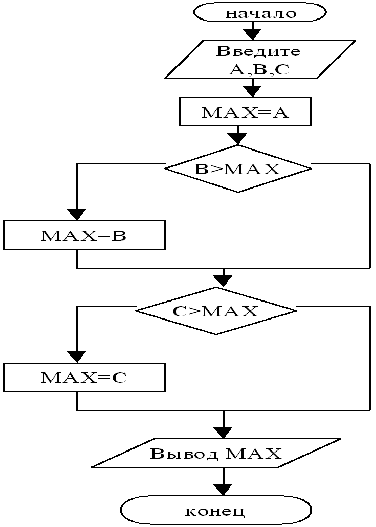


Задания:

1. *С помощью операторов data и read введите массив из семи слов. Составьте алгоритм и программу.*
2. *Заполните массив элементами последовательности Фибоначчи.*

Решение различных логических задач при обработке массивов связано с поиском различных элементов в массиве, с упорядочением массива по определенному принципу, с перестановками и заменами различных элементов в массиве и.т.д.

Решим задачу по нахождению max числа из трех действительных чисел. На примере этой задачи понятен сам принцип, который в дальнейшем будет применяться при нахождении min и max элементов в одномерных массивах.



Пример

Найдите max их трех чисел А, В, С.

10 INPUT “Введите числа А,В,С”;A, B, C

20 MAX=A

30 IF B>MAX THEN MAX=B

40 IF C>MAX THEN MAX=C

50 PRINT “MAX=”;MAX

60 END

**Пример**

Найдите MIN элемент одномерного массива A(N).

10 INPUT N

20 FOR I=1 TO N

30 INPUT A(I)

40 NEXT I

50 MIN=A(1)

60 FOR I=2 TO N

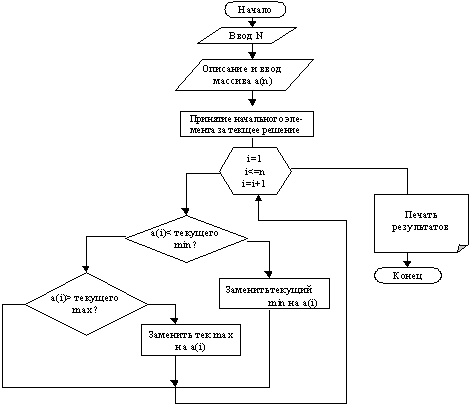
70 IF A(I)<MIN THEN MIN=A(I)

80 NEXT I

90 PRINT “MIN=”;MIN

**Пример**

В одномерном массиве A(N) найдите минимальный и максимальный элементы и их индексы.



10 REM min и max элемент массива

20 INPUT ”введите размер массива”;n

30 DIM a(n)

40 REM ввод массива

50 FOR i=1 TO n

60 INPUT a(i)

70 NEXT i

80 m1=a(0)

90 i1=0

100 m2=a(0)

110 i2=0

120 FOR i=1 TO n

130 IF a(i)>= m1 THEN 160

140 m1=a(i)

145 i1=i

150 GOTO 190

160 IF a(i)<= m2 THEN 190

170 m2=a(i)

180 i2=i

190 NEXT i

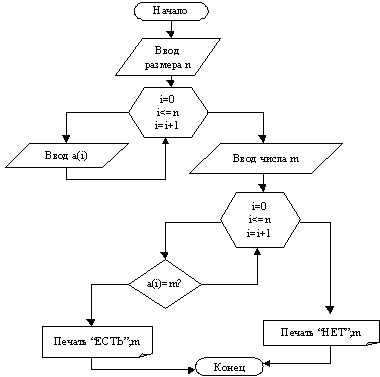
200 ? ”min=”;m1;”инд=”;i1

210 ?” max=”;m2;”инд=”;i2

220 END

**Пример**

*В одномерном массиве найти некоторое заданное число М.*



10 REM поиск заданного элемента

20 INPUT n

30 DIM a(n)

40 FOR i=1 TO n

50 INPUT a(i)

60 NEXT i

70 INPUT ”Введите число”; m

80 FOR i=1 TO n

90 IF a(i)=m THEN 130

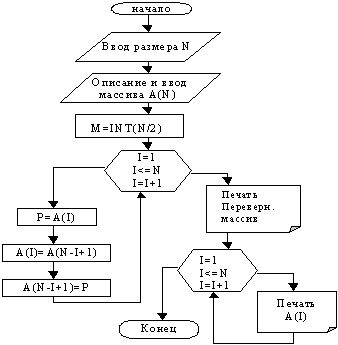
100 NEXT i

110 PRINT “НЕТ”;m

120 GOTO 140

130 PRINT ”ЕСТЬ”; m

140 END



**Пример**

*Изменить порядок следования элементов в массиве на обратный.*

10 INPUT “Введите размер”;N

20 DIM A(N)

30 FOR I=1 TO N

40 INPUT A(I)

50 NEXT I

70 M=INT(N/2)

80 FOR I=1 TO M

90 P=A(I)

100 A(I)=A(N-I+1)

110 A(N-I+1)=P

120 NEXT I

130 ? “Перевернутый массив”

140 FOR I=1 TO N

150 PRINT “A(“;I:”)=”;A(I)

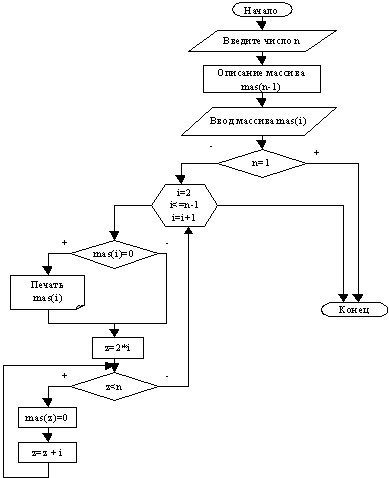
160 NEXT I

170 END

Пример

*Найдите все простые числа меньшие n.*

Простыми называются те числа, которые делятся сами на себя и на единицу. В предлагаемом сейчас алгоритме используется “просеивание” массива из i натуральных чисел.



CLS

INPUT “Введите число n”; n

DIM mas(n-1)

FOR i=1 TO n-1

mas(i)=i

NEXT i

CLS

PRINT”Простые числа меньше”; n

IF n=1 THEN END

FOR i=2 TO n-1

IF mas(i)<>0 THEN PRINT mas(i);

z=2\*i

WHILE z<n

mas(z)=0

z=z+i

WEND

NEXT i

END

**Пример**

*Описать числовые массивы X, Y, Z и выполните следующие преобразования:*

1. *переписать элементы массива X в массив Y в обратном порядке;*
2. *сформировать массив Y(1)=X(1)+X(N), Y(2)=X(2)+X(N-1), Y(3)=X(3)+ X(N-2) и т.д (N – четное);*
3. *записать в массив Y номера элементов массива X, лежащих на отрезке [0,1];*
4. *записать в массив Y элементы массива X, имеющие четные индексы, а в массив Z элементы, имеющие нечетные индексы;*
5. *записать в начало массива Y положительные, а в конец – отрицательные элементы массива X, сохранив порядок элементов.*

***1)***

10 INPUT ”Введите кол-во элементов”;N

20 DIM X(N), Y(N)

30 FOR I=1 TO N

40 PRINT ”введите элемент X(“ ;I: ”):”

50 INPUT X(I)

60 Y(N-I+1)=X(I)

70 NEXT I

80 FOR I=1 TO N

90 PRINT Y(I)

100 NEXT I

110 END

***2)***

10 INPUT N

20 DIM X(N), Y(N/2)

30 FOR I=1 TO N

40 INPUT X(I)

60 NEXT I

70 FOR I=1 TO N/2

80 Y(I)=X(I)+X(N-I+1)

90 PRINT Y(I)

100 NEXT I

110 END

***3)***

10 INPUT N

20 DIM X(N), Y(N)

30 K=0

40 FOR I=1 TO N

50 INPUT X(I)

60 IF X(I) > 1 AND X(I) < 0 THEN 100

70 K=K+1

80 Y(K)=I

100 NEXT I

110 FOR I=1 TO K

120 PRINT Y(I)

130 NEXT I

140 END

***4)***

10 INPUT N

20 DIM X(N), Y(INT(N/2)), Z(INT(N/2)+1)

30 K=0

35 L=0

40 FOR I=1 TO N

50 INPUT X(I)

70 IF I/2=INT(I/2) THEN K=K+1 : Y(K)=X(I) ELSE L=L+1: Z(L)=X(I)

80 NEXT I

90 FOR I=1 TO K

100 PRINT Y(I)

110 NEXT I

120 FOR I=1 TO L

130 PRINT Z(I)

140 NEXT I

150 END

***5)***

10 INPUT N

20 DIM X(N), Y(N)

30 K=0

40 FOR I=1 TO N

50 INPUT X(I)

70 IF X(I) > 0 THEN K=K+1 : Y(K)=X(I)

80 NEXT I

90 FOR I=1 TO N

100 X(I) < 0 THEN K=K+1 : Y(K)=X(I)

110 NEXT I

120 FOR I=1 TO K

130 PRINT Y(I)

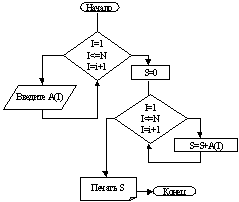
140 NEXT I

150 END

При обработке массивов приходится решать три типа задач: задачи на вычисление, логические задачи, связанные с сортировкой массива и объединенный тип задач. Рассмотрим несколько характерных задач на вычисление.

Пример

*В одномерном массиве A(N) вычислите сумму его элементов. Составьте алгоритм и программу.*



10 INPUT”Введите размер массива”:N

20 DIM A(N)

30 FOR I=1 TO N

40 INPUT A(I)

50 NEXT I

60 S=0

70 FOR I=1 TO N

80 S=S+A(I)

90 NEXT I

100 PRINT “S=”;S

**Пример 1**

*Найдите произведение элементов одномерного массива.*

Программа этой задачи аналогична за исключением следующего фрагмента

60 P=1

70 FOR I=1 TO N

80 P=P\*A(I)

90 NEXT I

100 PRINT “P=”;P

**Пример 2**

*Задан одномерный массив A(N) и число T. Найдите количество элементов массива, меньших Т*.

10 INPUT”Введите количество элементов массива”;N

20 INPUT”Введите число”; T

30 DIM A(N)

40 K=O

50 FOR I=1 TO N

60 PRINT”Введите A(“;I;”)”

70 INPUT A(I)

80 IF A(I)< T THEN K=K+1

90 NEXT I

100 PRINT”В массиве”;K;”элем меньших”;T

110 END

**Пример 3**

*Для одномерного массива A(N), найдите сумму элементов больших числа Т.*

10 INPUT “Введите размер массива”;n

20 INPUT “Введите число Т”;t

30 DIM a(n)

35 s=0

40 FOR i=1 TO n

50 INPUT a(i)

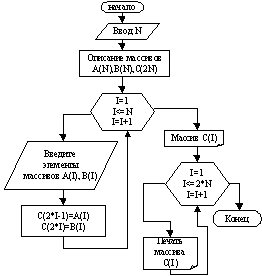
60 IF a(i) > t THEN s=s+a(i)

70 NEXT i

80 PRINT ”Сумма элементов массива, больших”; t; ”равна”; s

90 END

**Пример 4**



*Объедините два массива A(N) и B(N) в один массив C(2\*N) следующим образом: С(1)=А(1), С(2)=В(1), С(3)=А(2), С(4)=В(2)….C(2\*N-1)=A(N), C(2\*N)=B(N).*

10 INPUT ”Введите количество элм. В массивах А и В”;N

20 DIM A(N), B(N), C(2\*N)

30 FOR I=1 TO N

35 ? ”Введите A(“; I; “)”

40 INPUT A(I)

50 ? “Введите B(“; I; “)”

60 INPUT B(I)

70 C(2\*I – 1)=A(I)

80 C(2\*I)=B(I)

90 NEXT I

100 PRINT ”Вывод массива С”

110 FOR I=1 TO 2\*N

120 PRINT “C(“; I; “)=”; C(I)

130 NEXT I

140 END

**Пример 5**

*Для массива Х найдите:*

* + 1. *среднее арифметическое положительных элементов массива*
    2. *b) сумму элементов, стоящих на четных местах;*
    3. *c) среднее арифметическое элементов, лежащих в отрезке [1;2];*
    4. *d) количество положительных, отрицательных и равных нулю элементов;*
    5. *e) количество элементов, для которых ближайшим целым является 1.*

a)

10 INPUT ”введите кол-во элементов”;n

20 DIM x(n)

30 k=0: s=0

35 FOR i=1 TO n

40 PRINT ”введите x(“;i;”):”

50 INPUT x(i)

60 IF x(i)>0 THEN k=k+1 : s=s+x(i)

70 NEXT i

80 IF k>0 THEN PRINT ”сред. арифмет – “s/k ELSE PRINT ”нет положительных элем.”

90 END

b)

10 INPUT n

20 DIM x(n)

30 s=0

40 FOR i=1 TO n

50 ? ”Введите X(“; i;”)”

60 INPUT x(i)

70 IF (i/2)=INT(i/2) THEN s=s+x(i)

80 NEXT i

90 ? ”Сумма элементов на четных местах”;s

100 END

c)

10 INPUT ”введите кол-во элементов”;n

20 DIM x(n)

30 k=0 : s=0

40 FOR i=1 TO k

60 INPUT x(i)

70 IF x(i)<1 THEN 90

80 IF x(i)<=2 THEN k=k+1 : s=s+x(i)

90 NEXT i

100 IF k>0 THEN ? ”сред. арифм. на отрезке [1;2]-“;s/x ELSE ? ”нет элементов попадающих в [1;2]”

110 END

d)

10 INPUT n

20 DIM x(n)

30 k=0 : k1=0 : k2=0

40 FOR i=1 TO n

50 INPUT x(i)

60 IF x(i)>0 THEN k=k+1 ELSE k2=k2+1

80 NEXT i

90 PRINT ”В массиве”

100 PRINT ”Положительных элементов-“;k

110 PRINT ”Отрицательных элементов”;k2

130 END

e)

10 INPUT n

20 DIM x(n)

30 k=0

40 FOR i=1 TO n

50 INPUT x(n)

60 IF x(i)>1.5 AND x(i)>0.5 THEN k=k+1

90 NEXT i

100 PRINT k “элементов массива лежат в интервале [0,5;1,5]

110 END

*Дан одномерный массив A(N). Необходимо:*

1. *переписать массив в обратном порядке;*
2. *сжать массив, отбросив повторяющиеся элементы*
3. *сжать, отбросив каждый второй элемент.*

**Ввод элементов массива** может осуществляться несколькими способами:

1) DIM A (5)

A(0)=4: A(I)=-2.5: (3)=10: A(4)=0: A(5)=6: A(2)=3

Этот способ ввода удобен, когда массив небольшой

2)Ввод элементов массива с клавиатуры

DIM А(9)

FOR I=0 TO 9

INPUT A(I)

NEXT I

3) DIM B (6)

DATA 0, -2, 1, 5, 8, 5, 6. 26

FOR N=0 TO 6

READ B (N)

NEXT N

**Вывод элементов массива** можно осуществлять в строку и в столбик:

1) DIM A(3)

PRINT A(0), A(I), A(2), A(3)

Этот способ вывода удобен, когда массив небольшой

2) DIM A(10)

FOR I=0 TO 9

PRINT A(I)

NEXT I

**Основные приемы при решении задач с применением массивов**

*Пример 1.* Дан массив из 10 элементов. Найти сумму элементов массива

DIM A (9)

S=0

FOR N=0 TO 9

S=S+A(N)

NEXT N

PRINT ″сумма=″;S

*Пример 2.* Найти минимальный элемент массива и индекс этого минимального элемента

DIM L (9)

MIN=L(0): K=0

FOR I=1 TO 9

IF MIN>L(I) THEN MIN=L(I):K=I

NEXT I

PRINT ″минимум″; MIN

*Пример 3.* Даны 2 массива А(9), В(9). Получить массив, каждый элемент которого равен сумме соответствующих элементов данных массивов

DIM A(9), B(9), C(9)

FOR I=0 TO 9

C(I)=A(I)+B(I): PRINT C(I)

NEXT I

# Сортировка массива по возрастанию

|  |  |
| --- | --- |
| *1 СПОСОБ (метод пузырька)*  DIM A(9)  CLS  FOR i = 0 TO 9  INPUT A (i)  NEXT i  FOR i = 0 TO 9  FOR y = 9 TO i STEP – 1  IF A (y) >A(y-1) THEN GOTO 20  m = A (y)  A (y) = A (y-1) : A (y-1) = m  20: NEXT y  NEXT i  FOR i=0 TO 9  PRINT А(i)  NEXT i | *2 СПОСОБ (метод сортировки простым способом).*  DIM A(9)  PRINT “Введите 10 элементов”  FOR i=0 TO 9  INPUT A(i)  NEXT i  FOR i=0 TO 9  P=A(i) : K=i  FOR y=i+1 TO 9  IF A(y)>P THEN GOTO 50  P=A(y):K=y  50: NEXT y  A(K)=A(i):A(i)=P  NEXT i  PRINT “Отсортированные элементы”  FOR i=0 TO 9  PRINT A(i)  NEXT i  END |

*3 способ (метод простого включения)*

DIM A (10), В(9)

PRINT “Введите 10 элементов”

FOR i=0 TO 9

INPUT B (i)

NEXT i

FOR i=1 TO 10

A(i)=B(i-1)

NEXT i

FOR i=1 TO 10

P=A(I):Y=I-1

40: IF P<A(Y) AND Y>0 THEN A(Y+1)=A(Y):Y=Y-1:GOTO 40

A(Y+1)=P

NEXT i

PRINT “Отсортированные элементы”

FOR i=1 TO 10

PRINT A (i)

NEXT i

END

**Задачи на использование цикла при обработке массивов.**

1. Вывести положительные элементы массива Х(k), затем отрицательные элементы массива У(m) и количество выведенных чисел.
2. Даны k и массив X(k). Найти сумму.

(X1-P)2+(X2-P)2+...+(Xk-P)2, где Р=Х1+Х2+...+Хk)/k.

1. Даны m и массив Х ( m ). Найти произведение.

(2+Х21)\*(2+Х22)\*...\*(2+Х2m).

**4**. Найти номер элемента массива Х(k), ближайшего к С, где

С=(Х1+Х2+...+Хk)/k

1. Даны С, m и массив Т(m). Найти число элементов массива Т, меньших С, а для элементов, больших С, найти их среднее арифметическое.
2. Для массива Т(m) найти число элементов, больших предыдущего элемента, а для тех, которые меньше, найти их среднее арифметическое.
3. Найти номер наименьшего положительного элемента массива Х(m).
4. Для массива Х(m) напечатать сумму, произведение и номера положительных элементов после последнего нулевого элемента.
5. Даны k и массив Т(k). Найти сумму и количество элементов в массиве Т после первого нулевого элемента.
6. Даны m и массив А (m). Напечатать сумму отрицательных элементов массива А после первого нулевого элемента.
7. Даны m, координаты ХА,УА пункта А и в массиве К из 2m чисел координаты Х1, У1, Х2,У2,...,Хm,Ym пунктов В1, В2,...,Вm. Вывести номер и координаты пункта В наиболее удаленного от пункта А.
8. Найти общее количество нулевых элементов в массивах Х(m), У(k).
9. Образовать и вывести массив Т из неотрицательных элементов массива Х(m). Напечатать число элементов в массиве Т.
10. Вывести четные по значению среди положительных элементов массива Х(m), начиная с первого положительного элемента.
11. Удвоить наибольший элемент массива Х(m) и напечатать измененный массив.
12. Вывести ненулевые элементы массива Х(m) и их произведение.
13. Найти сумму и количество нечетных по значению элементов массива Х(m).
14. Образовать массив Т из ненулевых элементов массива Х(m). Затем вывести массив Т и количество элементов в нем.
15. В массиве Т(m) заменить нулевые элементы наибольшим элементом.
16. В массиве Х(m) найти сумму и количество элементов между первым и последним нулевым элементами.
17. Поменять местами наибольший и наименьший элементы массива Т(m).
18. В массиве Х(m) найти общее число элементов до первого и после последнего нулевых элементов.
19. Сколько элементов в массиве Х(m) после второго нулевого элемента.
20. Из элементов массивов Х(m) и У(m) образовать массив

Н(Х1,У1,Х2,У2,..., Х(m), У(m) из 2m элементов.

1. Округлить элементы массива Х(m) до ближайшего целого.
2. Из элементов массивов Х(m), У(k) образовать массив

Н((Х1,Х2,Х3,..., Хm, У1,У2,У3,...,Уk) из m+k элементов.

1. Найти номер второго нулевого элемента массива Х(m).
2. Найти номер предпоследнего нулевого элемента массива Х(m).
3. Переставить элементы массива Х(m) в обратном порядке, т.е. Хm,...,X2,Х1.
4. В массиве Х(m) найти число элементов между первым и вторым нулевыми элементами.
5. Сколько в массиве Т(m) элементов, меньших суммы всех элементов?
6. В массиве Т(m) нулевые элементы заменить суммой всех элементов.
7. Образуют ли арифметическую или геометрическую прогрессию элементы массива Х(m)?
8. В массиве Т(m) заменить элементы с четными номерами суммой элементов с нечетными номерами.
9. Найти расстояние Р между массивами А(k) и В(k) по формуле:

Р2-(А1-В1)2 +(А2-В2)2+...+(Ак-Вк)2.

1. В массив Р записать сначала отрицательные элементы массива Х(k). затем положительные.
2. В массиве Х(k) найти сумму элементов и их количество до первой перемены знака.
3. Из элементов массива Х(k), попадающих в отрезок [A,B], составить массив М и вывести его.
4. На плоскости ХОУ даны k точек массивами координат Х(k), Y(k). Образовать массив номеров точек вне I четверти.
5. Образовать массив Н номеров нулевых элементов массива Х(k).
6. В одномерном массиве с четным количеством элементов (2\*N) находятся координаты N точек плоскости. Они располагаются в следующем порядке: х1,у1,х2,у2,х3,у3 и т.д. Определить минимальный радиус окружности с центром в начале координат, которая содержит все точки, и номера наиболее удаленных друг от друга точек.
7. В двумерном массиве N x M содержатся М ординат N графиков функций в точках 1,2,3 ,..., М. Определить, пересекаются ли эти графики в одной точке, и если да, то найти ее номер. Считать, что графики могут пересечься только в указанных точках.
8. В одномерном массиве содержится график движения электропроезда от Перми до Кунгура в виде:

первый элемент- время движения до первой остановки;

второй элемент- время стоянки на первой остановке;

третий элемент- время движения до второй остановки;

четвертый элемент- время стоянки на второй остановке;

и так далее ....

последний элемент- время движения от предпоследней остановки до Кунгура.

1. В одномерном массиве хранятся результаты социологического опроса. Написать программу для подсчета количества результатов, отклоняющихся от среднего не более чем на 7%.
2. От острова Буяна до царства славного Салтана месяц пути. Капитан корабля записывает в вахтенный журнал количества миль, пройденных за день. Написать программу, определяющую, в какую десятидневку пройден большой путь.
3. В массиве хранятся данные о температуре окружающего воздуха за месяц. Определить самую холодную декаду месяц.
4. Рост студентов двух групп, в которых соответственно 23 и 25 человек, размещен в двух массивах. Требуется определить средний рост студентов каждой группы, а также количество студентов в двух группах вместе, имеющих рост выше 180 см.
5. 20 покупателей магазина оценивают качество товара оценками 3,4,5. Определить сколько процентов составляют покупатели, поставившие 5,4 и 3 в отдельности от общего числа покупателей, участвовавших в экспертизе товара.
6. В соревновании принимали участие команды трех классов по 6 человек в каждой. Результаты каждой команды занесены в определенный массив в виде баллов, которыми судьи оценивали каждого участника. Определить команду- победителя.
7. Необходимо протестировать 15 абитуриентов для распределения их по группам для изучения иностранного языка: английский язык, французский язык, немецкий язык, начальное знакомство с иностранным языком. Составьте программу, подсчитывающую количество абитуриентов в каждой группе в зависимости от того, как ответил абитуриент на вопрос об изучении иностранного языка в школе.
8. В течение января месяца в доме не работала котельная, поэтому температура в квартирах была непостоянной. Определить количество дней, в течение которых жильцам посчастливилось иметь температуру не ниже 15 градусов, а также вычислить среднюю температуру воздуха в квартирах за этот месяц. Для решения задачи использовать массив.
9. 18 учащихся сдавали норму по метанию гранаты. Составьте программу, определяющую какой процент учащихся выполнил норму. Если известно, что норма 38 метров.
10. В каждый солнечный день улитка, сидящая на дереве, поднимается вверх на 2 см, а в каждый пасмурный день опускается вниз на 1 см. В начале наблюдений улитка находилась в 30 см. от земли. Дан массив из 30 элементов, содержащий сведения о том был ли день солнечным или пасмурным. Составьте программу определения места нахождения улитки к концу 30 дня.
11. В день вашего рождения тетя положила на ваше имя в сбербанк вклад в размере 1000 рублей. Каждый год к вкладу добавляется S% годовых. Составьте программу для вычисления:

а) через сколько лет вклад достигнет R рублей?

б) какой размер вклада будет через 15 лет после вложения?

# Подпрограммы

Иногда бывает удобно некоторые фрагменты программы представить в виде отдельных частей, по несколько раз используемых в разных частях программы. Такие части оформляются как подпрограммы. Подпрограмма записывается следующим образом :

**GOSUB N** - переход на строку N

**RETURN** - конец подпрограммы.

При использовании оператора GOSUB N компьютер запоминает следующий после GOSUB номер строки и после завершения подпрограммы начинает дальнейшее исполнение с этого номера.

Число подпрограмм не ограничено.

**Функция пользователя.**

В тех случаях, когда функция не является элементарной, или у функции несколько аргументов, удобно использовать функцию пользователя: **DEF FN.** Имя функции всегда должно начинаться с букв **FN**.

DEF FN не может быть рекурсивной (не может вызывать сама себя), и должна быть определена перед пользователем.

***Пример:***

*Пусть X и Y - натуральные числа. R - остаток от деления X на Y. Выразить R(X,Y) - R функция от X и Y.*

***R=X- INT(X/Y)\*Y -*** остаток от деления ***X*** на ***Y***

***DEF FN R(X,Y)=X-Y\*INT(X/Y)***

В этом примере ***R(X,Y)*** - функция двух аргументов.

***PRINT (SIN(47\*PI/180)-COS(78\*PI/180)^2)/(EXP(1.5)-SIN(13\*PI/180))***

# Работа с файлами

**Базы данных**

В некоторые программы надо многократно вводить одни и те же данные, что очень утомительно. Можно сэкономить много времени используя базы данных.

DATA список констант через запятую

READ список имен переменных

RESTORE номер строки с DATA- ой для восстановления

10 DATA 1,4,-5,10,56

30 FOR I=1 TO 5

40 READ X

50 PRINT X

60 NEXT I

70 END

В результате исполнения программы мы получим на экране :

1

4

-5

10

56

10 DATA 15,4,-5,80,56,105,8,9,4,6

20 CLS

30 FOR I=1 TO 5

40 READ A, B

50 PRINT A+B

60 NEXT I

70 END

А сейчас более подробно разберем эту программу. В строке 10 мы задали список констант, а затем стали их считывать. Сначала А, затем В, затем их сумму выводим на экран. И так 5 раз.

A | 15|-5|56|8|4|

B | 4|80|105|9|6|

После выполнения программы мы получим :

19

75

161

17

10

Константы могут быть и символьными, как работать с символами я расскажу чуть позже .

**Запись в файл и вывод из файла**

Иногда необходимо чтобы программа записала результаты вычислений в какой-то файл на диске, пусть это будет файл proba.txt .

Для начала этот файл надо открыть, и указать зачем, к примеру, открыть файл proba.txt для ввода. Сделать это можно так:

OPEN "proba.txt" FOR OUTPUT AS# 1

AS #1 - логический номер файла, это номер произволен, т.е. Вы сами можете его выбрать.

Ввод в файл осуществляется так: PRINT #N, A,B

Где N - логический номер файла (в нашем случае это 1), А и В- имена переменных записываемых в файл.

**INPUT #N,A,B**- вывод из файла с логическим номером N, значений переменных А и В.

**CLOSE #N**- закрытие файла с логическим номером N.

После того как Вы закончите работать с файлом его следует закрыть.

# Комбинированные типы.

Под **переменной** в языках программирования понимают программный **объект** (число, слово, часть слова, несколько слов, символы), имеющий имя и значение, которое может быть получено и изменено программой. При объявлении переменных можно указать тип данных. Это делается с помощью оператора **DIM**. В бейсике имеется несколько встроенных типов: числовые (integer, long, single, double) и стоковые (string, string \*).

При работе с большим числом данных одного типа очень удобно использовать массивы. Массив, это разновидность переменной. Он дает возможность хранить сколько угодно значений одного типа под одним и тем же именем. К каждому конкретному значению массива, необходимо обращаться через числовой индекс.

При написании программы, возникает необходимость описать **характеристики** (свойства) некоторого **объекта**, представляемого и обрабатываемого в программе. Таким объектом может быть человек, некоторый вычислительный комплекс, письмо, посылаемое по почте и т.д. Во всех подобных случаях свойства объекта представляются значениями различных типов и поэтому для их описания не могут быть использованы массивы.

Для описания объекта «ученик» могут понадобиться, например, следующие характеристики:

* фамилия, имя и отчество (строки);
* возраст (integer);
* пол (строка);
* класс (integer);
* буква класса (символ); и т.д.

Для представления такой разнородной, но логически связанной информации удобно использовать **комбинированный тип**. Необходимо отметить, что в данном случае определенные компоненты комбинированного типа, ввиду их различной природы, не могут идентифицироваться порядковыми номерами (индексами), как в массивах, поэтому для обозначения компонентов используются идентификаторы (имена). Таким образом, описание комбинированного типа представляет собой список описаний его элементов; каждое описание похоже на описание простой переменной. Для примера, приведенного выше, описание комбинированного типа PUPIL (ученик) может выглядеть следующим образом:

TYPE Pupil

fio AS STRING \* 20

age AS INTEGER

sex AS STRING \* 6

class AS INTEGER

classname AS STRING \* 1

END TYPE

Определив собственный тип данных, вы можете использовать его для объявления переменных этого типа.

DIM Schoolchildrens AS Pupil

DIM Group(1 TO 25) AS Pupil

Доступ к компонентам (свойствам) переменной пользовательского типа осуществляется путем указания точки после имени переменной.

Schoolchildrens.fio = "Иванов Иван"

Schoolchildrens.age = 15

Schoolchildrens.sex = "male"

Schoolchildrens.class = 10

Schoolchildrens.classname = "А"

PRINT Schoolchildrens.fio, Schoolchildrens.age, Schoolchildrens.sex, Schoolchildrens.class, Schoolchildrens.classname

***Пример простой программы:***

REM использование комбинированных типов

REM описание типа ученик

TYPE Pupil

fio AS STRING \* 20

age AS INTEGER

sex AS STRING \* 6

class AS INTEGER

classname AS STRING \* 1

END TYPE

REM объявление массива из 3 элементов типа ученик

DIM Group(1 TO 3) AS Pupil

Group(1).fio = "Иванов Иван"

Group(1).age = 15

Group(1).sex = "male"

Group(1).class = 10

Group(1).classname = "А"

Group(2).fio = "Петрова Маша"

Group(2).age = 14

Group(2).sex = "female"

Group(2).class = 10

Group(2).classname = "Б"

Group(3).fio = "Сидоров Вася"

Group(3).age = 16

Group(3).sex = "male"

Group(3).class = 11

Group(3).classname = "В"

REM выводим на экран учеников 10 класса

FOR i=1 TO 3

IF Group(i).class = 10 THEN PRINT Group(i).fio

NEXT i

END

# Операторы графики

Современные компьютеры обладают превосходными графическими возможностями - у современных мультимедийных ЭВМ возможен даже просмотр и редактирование видеофильмов. Графика ориентирована на построение простейших геометрических фигур и реализацию простейших возможностей динамической графики.

Для того чтобы работать с графической информацией необходимо установить графический режим работы специальным оператором и использовать соответствующие операторы:

1. **SCREEN N**- оператор включения графического режима, где N - номер режима работы (0 - текстовый режим работы, от 1 до 13 - графические режимы работы)
2. **COLOR** - оператор, задающий цвет текста и фона

**COLOR C1, C2**, где C1 задаёт цвет текста, а C2 - цвет фона. C1, C2=0-15.

|  |  |
| --- | --- |
| Число | Цвет |
| 0 | *Черный* |
| 1 | *Синий* |
| 2 | *Зеленый* |
| 3 | *Голубой* |
| 4 | *Красный* |
| 5 | Фиолетовый |
| 6 | Коричневый |
| 7 | Серый |
| 8 | Темно-серый |
| 9 | Светло-синий |
| 10 | Светло-зеленый |
| 11 | Светло-голубой |
| 12 | Светло-красный |
| 13 | Светло-фиолетовый |
| 14 | Желтый |
| 15 | Белый |

Некоторые сочетания цветов в операторе COLOR QBasic:

1) 14, 8 - желтый на темно-сером фоне;

2) 1, 7 - синий на светло-сером фоне;

3) 5, 14 - фиолетовый на желтом фоне;

4) 4, 11 - красный на голубом фоне;

5) 5, 10 - фиолетовый на светло-зеленом фоне;

6) 15, 5 - белый на фиолетовом фоне;

7) 14, 1 - желтый на синем фоне;

8) 15, 1 - белый на синем фоне;

9) 8, 11 - темно-серый на голубом фоне;

10) 1, 15 - синий на белом фоне.

Как создать движение по экрану окружности?

Очень просто. Надо использовать цикл.

CLS

FOR I=640 TO 1

SCREEN 12

CIRCLE (I,200),50,14

NEXT I

END

В этой программе координата X окружности равна I, а I меняется от 640 до 1.

1. **PSET (X,Y)** - оператор рисования точки с координатами Х, У (0<= X<=XК 0<=Y<= YК, где ХК и УК - размер экрана в текущем режиме работы)
2. **LINE (X1, Y1) - (X2, Y2)** - оператор рисования отрезка,

где (X1, Y1), (X2, Y2)- координаты концов отрезка, лежащих в диагональных вершинах

**LINE (X1, Y1) - (X2, Y2),N, B**- оператор рисования

прямоугольника со сторонами параллельными осям X2,Y2 координат, если вместо параметра B

используется BF, то прямоугольник будет закрашенный X1, Y1

N –

X1,Y1

X2,Y2

X1,Y1

X2,Y2

4. **CIRCLE (X, Y),R** - оператор рисования окружности

**CIRCLE (X, Y),R,N,A,B,K** - оператор рисования дуги от угла A до B против часовой стрелки, углы измеряются в радианной мере ( см. диаграмму).

N - цвет

K – коэффициент сжатия эллипса

A

K=

R1

B

R2

R

1. **PAINT (X, Y)** - оператор окрашивает

замкнутую область, начиная от указанной точки

В основном работаем в **SCREEN 12**, который имеет следующие характеристики:

черно- белый экран,

размер по горизонтали 0<=X<=640,

по вертикали 0<=Y<=480.

*Пример 1.*Постройте три точки, которые являются вершинами равнобедренного треугольника.

CLS

SCREEN 2

PSET (100,100)

PSET (200,100)

PSET (150,200)

END

*Пример 2.* Постройте главную и побочную диагонали экрана.

CLS

SCREEN 12

FOR X=0 TO 639

Y=-425/600\*X+460

PSET (X, Y)

NEXT X

END

*Пример 3.* Постройте n концентрических окружностей

INPUT N

SCREEN 2

CLS

FOR X=1 TO N

R=1+3.14\*(X -1)

CIRCLE (315, 100) ,R,

NEXT X

END

*Пример 4*. Постройте 10 окружностей, расположенных случайным образом, пользуясь функцией RND.

CLS

SCREEN 2

FOR I=1 TO 10

CIRCLE (RND(1) \* 639, RND(1)\*190), RND(1)\*100,0,0.14

NEXT I

END

*Пример 5.* Составить программу, изображающую окружность в закрашенном прямоугольнике

SCREEN 2

CLS

LINE (60, 30) - (180, 150),,B

CIRCLE (120, 90), 60

PAINT (61, 31)

PAINT (179, 149)

END

**Задачи на построение изображений.**

Графики элементарных функций

Все рисунки, которые до сих пор были нарисованы на экранах, составлялись только из точек, отрезков прямых и дуг окружностей. Часто приходится в инженерных расчетах строить на экране дисплея любую кривую, для которой существует аналитическая (записанная в виде формулы) зависимость между абсциссой и ординатой каждой ее точки.

Рассмотрим кривую, которая задана уравнением у=22+1. Это обыкновенная парабола, вершина которой лежит на единицу выше оси абсцисс. Для построения параболы у=х2+1 в требуемом интервале изменения переменной надо работать по меньшей мере в двух четвертях координатной плоскости.

Чаще всего бывает нужна система координат с началом центра экрана (как на математической плоскости). В этом случае, установив начало координат в требуемой точке (х0,у0) на экране, можно координаты (х,у) произвольной точки кривой определять разностью (х-х0,у-у0). После этого в программе можно употреблять не только положительные, но и отрицательные значения.

Рисунок получается маленький, почти незаметный, поэтому увеличить масштаб изображения: надо увеличить рисунок по х и по у в 319 раз, так как графический экран имеет размеры 640 на 480. Разделив 640 пополам, получим 320, т.е. от 0 до 319 приходится на положительную и отрицательную оси абсцисс. На экране можно получить ограниченное количество значений х, то все их легко перебрать с помощью цикла: Начальное значение цикла х=1, конечное значение х=1, а шаг по х при 639 значениях х на отрезке [-1,1] получается равным 2/639 или 1/319.

**Примеры решения задач**

*Пример* 1.Постройте график функции у=2х+4

SCREEN 2

LINE (320,0)-(320,200)

LINE (0,100)-(640,100)

FOR x =-20 TO 20

y= (2\*x+4)\*(-1)

PSET (x+320, y+100)

NEXT x

*Пример 2.* Постройте график функции у=х3.

SCREEN 12

LINE (0,240)-(640,240),13

LINE (320,0)-(320,480),13

FOR x=-20 TO 20 STEP 0.01

y=x^3

PSET (320+4\*x, 240+y), 3

NEXT

*Пример 3.* Решите графически систему уравнений и проверьте, есть ли общее решение:

Решение:

SCREEN 12

LINE (320,500)-(320,10),9

LINE (0,100)-(620,100),9

FOR x=-10 TO 10 STEP 0.01

y=-x^2

PSET (320+4\*x, 160+y), 10

NEXT x

FOR x=-100 TO 100 STEP .01

y=-6-x

PSET (320+x, 160+y), 12

NEXT x

LINE (320,10)-(300,50),9

LINE (320,10)-(340,50),9

Графики пересекаются т.е. общее решение есть.

**Задачи по теме «Графики элементарных функций»**

1. Постройте график функции у= 1/2х+3
2. Постройте график функции у=х2+4
3. Решите графически систему уравнений и проверьте, есть ли общее решение:
4. Решите графически систему уравнений и проверьте, есть ли общее решение:
5. Решите систему уравнений и проверьте, есть ли общее решение:
6. Решите графически систему уравнений и проверьте, есть ли общее решение:
7. Постройте график тригонометрической функции у=сos(х).
8. Постройте график тригонометрической функции у= 2+sin (x).
9. Постройте график тригонометрических функций у=2+SIN(X/2).
10. Постройте график тригонометрической функции у= 1/2\*COS (X)
11. Постройте график тригонометрической функции Y=TAN(X)
12. Постройте график тригонометрической функции у=0,5\*COS(X)-1

***Экранные заставки*** — это динамические изображения, воспроизведение которых включается автоматически при отсутствии в течение заданного времени событий, вызванных пользователем. Первоначальное назначение заставок состояло в том, чтобы снизить угрозу «выгорания люминофора» на тех участках экрана, которые подвержены особо длительному стационарному воздействию электронного луча. Результатом этого эффекта было образование бурых пятен в местах длительного воздействия луча.

Современным мониторам эффект «выгорания люминофора» не грозит, но экранные заставки продолжают использовать как средство сокрытия экранной информации от посторонних наблюдателей в период отсутствия владельца компьютера на рабочем месте.

Заставки в основном используют вывод простых геометрических фигур с анимацией. Сегодня мы попробуем смоделировать работу заставки, используя графические процедуры в Бейсике.

Обычно вывод происходит текстовой информации. А нам нужен вывод графической информации (в пикселах). Графический режим характеризуется разрешением (количество отображаемых пикселей на дюйм), количеством цветов и количеством видеостраниц (при работе с анимацией).

А теперь перейдем к самостоятельной работе за компьютером: будем последовательно находить информацию о графических процедурах, используя справочную систему Qbasic (используется информация из русифицированного файла справки Qbasic.hlp). Для этого в среде Qbasic выберите пункт меню Help (Помощь) – > Index (Предметный указатель), или нажмите сочетание клавиш Alt+F1. Затем будем выполнять задания по найденному материалу. По окончании работы вы должны получить программу, имитирующую работу экранной заставки.

**Некоторые полезные команды QBasic.**

Оператор SWAP X1, X2 - заставляет переменные X1 и X2 обменяться своими значениями. Пример:

X = 3

Y = 5

SWAP X, Y

Директива FILES выдает список подкаталогов и файлов в текущем каталоге. Кроме того, выдается свободная память на жестком диске.

Директива SHELL осуществляет сеанс связи с MS-DOS. Возвращение обратно из DOS в QBasic - команда EXIT.

Кроме того, директива SHELL "name" позволяет запускать программу name.exe или name.com, указанную в кавычках. Эта программа должна находиться в текущем каталоге, там же, где и \*.bas файл. Расширение exe или com можно не указывать.