Алгоритмизация и

программирова ние

Ha QBasic





Полезные советы

Для глубокого понимания очень полезно «прокручивать» все шаги компьютера

- у себя в голове,
- следует уметь *читать* коды программ. Каждую лекцию и ,
 - на первый взгляд, хорошо усвоенную программу
 - обязательно необходимо закреплять
 - практической работой на компьютере.
- Иначе научиться программировать просто невозможно.

ВВЕДЕНИЕ

- Basic(Beginner's All-purpose Symbolic Instruction Code)один из первых языков высокого уровня был разработан в начале 60-х годов в Дартмудском колледже для учебных целей .Это язык-долгожитель, число версий его не поддаётся пересчёту.
- Ваѕіс язык программирования, на котором Билл Гейтс, будучи в возрасте 13-ти лет, написал свою первую программу для игры в крестики-нолики. Этот язык для главы корпорации Microsoft явился трамплином к тому, чтобы стать и лидером наиболее современных компьютерных технологий, и одним из самых богатых людей в мире. Билл Гейтс сам заработал свои капиталы исключительно благодаря своему уму, дальновидности и предприимчи вости.
- Важным шагом стала версия языка Quick Basic, реализованная в QBasic и Visual Basic.

ВВЕДЕНИЕ

- Большинство инженеров и научных сотрудников сходятся во мнении, что *одинаково хорошо* можно выполнить программирование как на языке Turbo Pascal, так и на языке QBasic.
- Удалось сгладить многие недостатки QBasic,
 обусловленные длительным периодом его развития (более
 40 лет), в этом принимали участие большое количество спе
 циалистов, принадлежащих разным поколениям.
- Знать QBasic оказывается полезным, поскольку на нем пишутся DOS-инструкции для конфигурирования компьютера, подпрограммы в многочисленных приложениях Windows. Хорошо зная одну, базовую версию Бейсика QBasic, в дальнейшем можно свободно переходить к работе с другими версиями этого популярного языка.

ВВЕДЕНИЕ

- В наше время популярным инструментом разработки приложений Windows является язык визуального программирования Visual Basic . Поэтому перед изучением Visual Basic разумно изучить вначале QBasic, который является его составной частью.
- Освоить QBasic должен, пожалуй, каждый желающий научиться программировать, ведь он является распространенным языком программирования. Такова действительность QBasic завоевал мир, сделал он это всерьёз и надолго.
- Для глубокого понимания очень полезно «прокручивать» все шаги компьютера у себя в голове, следует уметь читать коды программ. Каждую лекцию и , на первый взгляд, хорошо усвоенную программу обязательно необходимо закреплять практической работой на компьютере. Иначе научиться программировать просто невозможно.

1.Элементы алгоритмизации

1.1.Понятие алгоритма. Свойства алгоритмов.

- Слово «алгоритм» появилось как результат латинской транскрипции имени великого ученого IX в. Мухаммеда ибн Мусы Аль-Хорезми, который сформулировал общие правила (алгоритмы) выполнения арифметических операций над десятичными числами.
- Алгоритм это определенная последовательность действий (команд ,шагов), чёткое предписание конкретному исполнителю , выполнение которого приводит к достижению поставленной цели.
 - (существуют и другие формулировки).
- Основные свойства алгоритма: дискретность, определенность, результативность, массовость.
- Программирование это реализация заданного алгоритма на формальном языке программирования.
 Программирование позволяет переложить проведение и анализ информационных процессов на современную вычислительную технику.

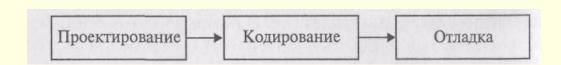
1.2.Способы представления алгоритмов.

Алгоритм может быть задан способами:

- 1) на естественном языке ;
- 2) аналитически (формулой);
- 3) графически (в виде блок-схемы);
- 4) на алгоритмическом языке(РАЯ);
- 5) на языке программирования.

1.3.Основные структуры алгоритмов.

- Основные структуры алгоритмов это ограниченный набор блоков и стандартных способов их соединения для выполнения типичных последовательностей действий.
- Структурный подход предполагает использование только нескольких основных структур (линейных, ветвящихся, циклических), комбинация которых дает все многообразие алгоритмов и программ.
- В процессе изготовления программного продукта программист должен пройти определенные этапы.



Этапы решения задачи на пк

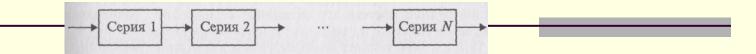
- Процесс решения задачи на компьютере состоит из этапов:
- I. Постановка задачи.
- II. Математическая модель.
- III. Алгоритмизация задачи.
- IV. Программирование.
- V .Отладка задачи на компьютере.
- VI .Анализ результата.

Базовые алгоритмические структуры

- В конце 60-х начале 70-х гг. ХХ столетия появляется дисциплина, которая получила название структурного программирования.
- Её развитие связано с именами Э.В. Дейкстры , X.Д. Миллса , Д. Е. Кнута и других ученых. Структурное программирование до настоящего времени остается *основой* технологии программирования. Соблюдение его принципов позволяет программисту быстро научиться писать ясные , безошибочные, надёжные программы.
- В основе структурного программирования заложена теорема, которая была строго доказана в теории программирования. Она утверждает, что алгоритм для решения любой логической задачи можно составить только из структур «следование, ветвление, цикл».
- Их называют базовыми алгоритмическими структурами.

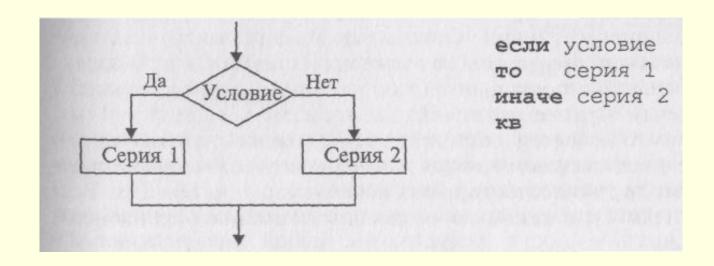
1.3.1. Следование

■ Следование — это линейная последовательность действий:



1.3.2. Ветвление

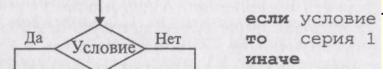
Ветвление — алгоритмическая альтернатива. Управление передается одному из двух блоков в зависимости от истинности или ложности условия. Затем происходит выход на общее продолжение:



Неполная форма ветвления

Неполная форма ветвления имеет место, когда на одной ветви

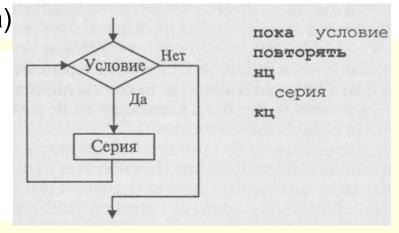
пусто:

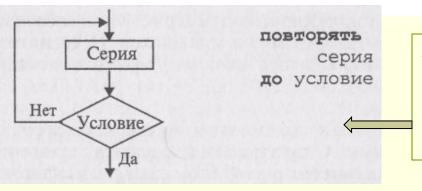


1.3.3. Цикл

Цикл — повторение некоторой группы действий по условию. Различаются следующие типы цикла:

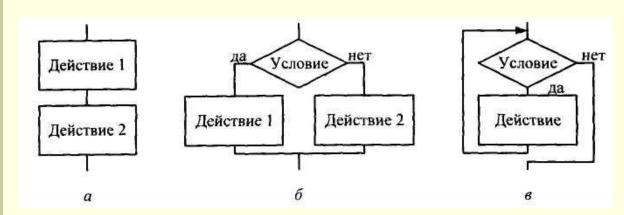
Цикл с предусловием (цикл-пока)





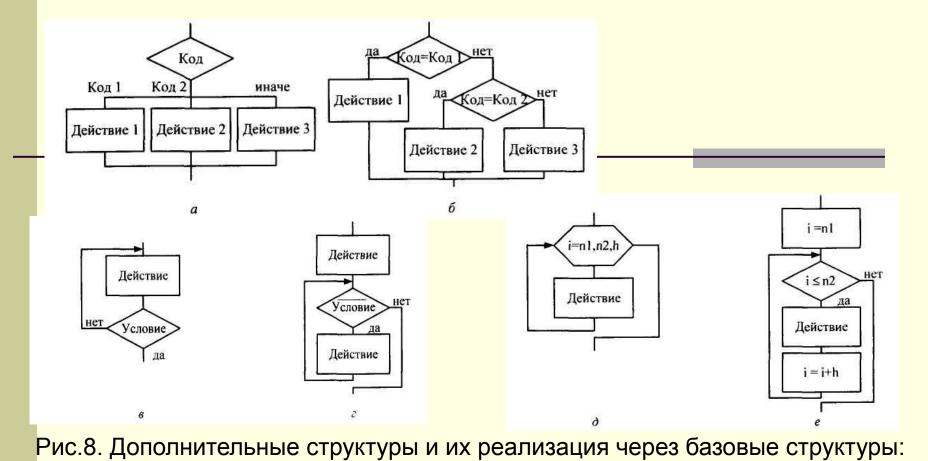
Здесь тело цикла предшествует условию цикла. Тело цикла повторяет своё выполнение, если условие ложно. Повторение кончается, когда условие станет истинным.

Рассмотренные выше блок-схемы можно изобразить и так:



Базовые алгоритмические структуры: следование, ветвление, цикл- пока. Кроме базовых алгоритмических структур используют *дополнительные* структуры, производные от базовых:

- выбор обозначающий выбор одного варианта из нескольких в зависимости от значения некоторой величины (рис.8 а);
- *цикл-до* обозначающий повторение некоторых действий до выполнения заданного условия, проверка которого осуществляется после выполнения действий в цикле (рис8. в);
- цикл с заданным числом повторений (счетный цикл) обозначающий повторение некоторых действий указанное количество раз (рис. 8, д).



выбор *(а-б),* цикл-до *(в-г)* и цикл с заданным числом повторений *(д-е)*На рис. 8,6 ,г и е показано, как каждая из дополнительных структур может быть

реализована через базовые структуры.

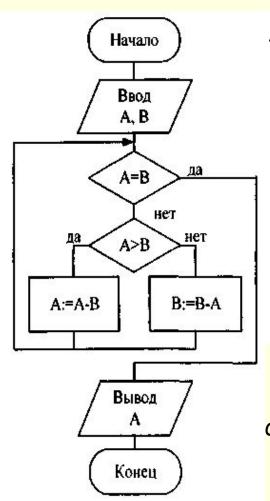
Перечисленные структуры были положены в основу *структурного программирования* - технологии, которая представляет собой набор рекомендаций по уменьшению количества ошибок в программах .

Сложный алгоритм состоит из соединенных между собой базовых структур. Соединяться эти структуры могут двумя способами: последовай последовай вложенным.

Пример 1.

- Разработать алгоритм вычисления наибольшего общего делителя двух натуральных чисел.
- Существует несколько способов нахождения наибольшего общего делителя двух натуральных чисел. Самым простым из них является так назы ваемый «алгоритм Евклида». Суть этого метода заключается в последовательной замене большего из чисел на разность большего и меньшего. Вычисления заканчиваются, когда числа становятся равными.
- На рис. 9. показана блок-схема алгоритма, а рядом приведено его описание на псевдокоде.

Алгоритм Евклида



Алгоритм Евклида:

Ввести А,В **цикл-пока А ≠ В если А > В**

то A := A - В

иначе В := В - А

все-если

все-цикл

Вывести А

Конец алгоритма.

Структурная методика алгоритмизации — это не только форма описания алгоритма, но это ещё и способ мышления программиста. Создавая алгоритм, нужно стремиться составлять его из стандартных структур. Если использовать строительную аналогию, можно сказать, что структурная методика построения алгоритма подобна сборке здания из стандартных секций в отличие от складывания по кирпичику.

1.4. Трансляторы, компиляторы, интерпретаторы.

- Составлять программу на языке высокого уровня, конечно, удобно.
 Набрал текст в редакторе, записал команды по алгоритму решения задачи и всё.
- Но текст программы это ещё не готовое приложение, это только набор символов, которые «запустить» в работу невозможно.
- Для того чтобы набранный на каком-то языке программирования исходный текст приложения ожил, заработал, смог выполнять заложенные в него команды, применяются так называемые трансляторы.
- Транслятор это специальная программа, которая автоматически превращает исходные тексты в работающее приложение, например калькулятор. Исходный текст, например, набор инструкций Бейсика, переводится непосредственно в машинный код при этом сразу формируется файл программы, отвечающий всем соглашениям операционной системы.
- Такой процесс перевода исходного текста в инструкции процессора и прерывания системы называется компиляцией, а программы, выполняющие этот процесс, — компиляторами.

2.Простейшие конструкции языка QBASIC.

2.1. Алфавит

- Совокупность допустимых символов образует алфавит: это буквы (прописные и строчные латинские буквы), цифры (арабские цифры то 0 до 9) и специальные символы(знаки операций, знаки пунктуации и зарезервированные слова).
- Между группами символов алфавита вставляются разделители.
- Из букв и цифр и знаков подчеркивания строятся идентификаторы.
- Написание прописных и строчных букв в идентификаторах не различается: слова VAR, vAR и VaR трактуются, как одно слово.
- Длина идентификатора может быть любой, но существенны только первые 63 символа.
- Знаки операций:
- +, -, *, /, \, ^ , >, <, <>, <= , >=, а также арифметические и логические функции (NOT, AND, OR, XOR, IMP и т. д.).

АЛФАВИТ

- Множество знаков пунктуации складывается из следующих символов:
- выделение комментария;
- () выделение индексов массивов, алгебраические скобки;
- ' апостроф;
- ; и, разделение списка ввода-вывода операторов PRINT и INPUT;
- тотделение нескольких операторов друг от друга в одной строке;
- = знак присваивания;
- разделение элементов списка; отделение целой части от дробной;
- % признак целого числа;
- & признак длинного целого числа;
- ! признак дробного числа;
- # признак дробного числа двойной точности;
- \$ признак символьной строки,
- ? приглашение оператора PRINT
- К зарезервированным словам относятся операторы, имена логических и арифметических функций. Список зарезервированных слов приводится в таблице.

2.2. Структура данных

- Под типом данных понимается множество допустимых значений переменных, а также совокупность операций над ними.
- В QBASIC можно выделить следующие группы типов:
- целые;
- вещественные;
- логические данные;
- массив;
- символьные данные;
- файлы.

2.2.1. Целые типы.

- В QBASIC введено два стандартных целых типа, которые отличаются форматами и диапазонами допустимых значений.
- Таблица1

Tun	Значение	Формат
% - INTEGER	-32768 32767	Знаковый
& - LONGINT	-21474836482147483647	Знаковый

ТАБЛИЦА 2

Для работы с целыми типами данных используются следующие арифметические функции, результат которых тоже целое число:

ABS(N)	Абсолютная величина N
A\B	Целая часть от деления
A MOD B	Остаток от деления
+,*,-	Сложение, умножение, вычитание
^	Возведение в степень
FIX (X)	Получение целой части вещественного числа Х
CINT(X)	Округление до целого вещественного числа Х
CLNG(X)	Округление до длинного целого вещественного числа X
INT(X)	Получение наибольшего целого числа, которое меньше или равно X

2.2.2. Вещественные типы.

В QBASIC определено два стандартных вещественных типа, которые отличаются форматами и диапазонами допустимых значений:

		Таблица3
Tun	Значение	Число цифр
!-REAL	-2.9*10-38.1.7*1038	32
#- DOUBLE	-2.9*10-381.7*Ю38	64

Результат работы функций сложения, умножения, деления, вычитания и возведения в степень вещественных чисел дает вещественное число.

Кроме этого, используются следующие функции, результат которых - вещественное число:

ТАБЛИЦА 4

Функция	Назначение	
ABS(X)	Абсолютное значение Х	
ATN(X)	Арктангенс Х	
COS(X)	Косинус Х	
SIN(X)	Синус Х	
TAN(X)	Тангенс Х	
EXP(X)	Ex	
LOG(X)	Натуральный логарифм	
SQR(X)	Квадратный корень Х	
RND(X)	Получение случайного числа	
CDBL(X)	Представление числа с двойной точностью	
CSNG(X)	Представление числа с одинарной точностью	25

2.2.3. Логические данные.

Логические данные, которые имеют значение либо "истина" - 1, либо "ложь" - 0, обрабатываются с помощью логических опе раций и операций сравнения. К ним относятся:

- NOТ отрицание;
- OR объединение, или логическое сложение;
- AND пересечение, или логическое умножение;
- XOR исключающее ИЛИ или сложение по модулю два;
- EQV эквивалентность;
- IMP импликация, или следование. Результат работы операций задается следующей таблицей истинности (таблицы значений):
- XOR исключающее ИЛИ или сложение по модулю два;
- EQV эквивалентность;
- ІМР импликация, или следование.

Результат работы операций задается следующей таблицей истинности (таблицы значений):

Запомнить таблицу можно ,помня следующее:

- операция отрицания меняет значение операнда на противоположное;
- ¬для того чтобы результат операции логического умножения был истинен, все операнды должны быть истинны;
- ¬для того чтобы логическое сложение дало истину, нужно, чтобы хотя бы один операнд был истинен,
- □ а сложение по модулю два дает истину, только если операнды имеют разные значения; иногда эту операцию называют выбором альтернативы (или жегалкинским сложением).
- Для определения результата операции эквивалентности нужно помнить, что он истинен, если операнды равны между собой, и ложен в противном случае. Для операции импликации из ложного операнда следует все что угодно результат будет истинен, а из истины следует только истина.
- При обработке логических данных используются также операции срав нения, которые называются логическими отношениями:
- =, < >, X, >, <, <= ,>=.
- Результат операции логического отношения равен минус единице, если задаваемое отношение выполняется, и нулю, если условие ложно.

2.2.4. Массив

- Представляет собой заранее известное количество однотипных элементов, снабженных индексами. Массив может быть одномерным или многомерным.
- Чтобы задать массив, необходимо использовать зарезервированное слово **DIM**, значения индексов массива и тип элементов массива.
- Например, объявление одномерного массива 11 целых чисел может быть задано так: DIM N%(10)
- Обратите внимание, что интерпретатор устанавливает минимальное значение индекса, равное нулю.
- Например, запись DIM BB(5,7), X(5)
- объявляет двумерный массив ВВ из 48 чисел обычной точности и одномерный массив из шести таких чисел.

Массив

- Оператор OPTIN BASE устанавливает минимальное значение индексов массива. Он должен быть указан до объявления массивов:
- OPTIN BASE n,
- где n равно единице или нулю.
- Оператор ERASE *отменяет* объявление массивов, сделанных оператором DIM:
- ERASE список имен массивов.

2.2.5. Символьные данные.

- В QBASIC имеется ещё тип данных, который называется символьным. Для того чтобы показать, что вы используете переменную такого типа, необходимо в имени этой переменной справа записать знак \$:
- например, A\$, DF\$, STR\$.
- Этот тип является порядковым, и значения символьных переменных можно сравнивать между собой (>, <, >=, <=):
- например ,"vit" < "vita"<"vitaon".</p>
- Возможна между ними и конкатенация (+): "мото" + "цикл"
 "мотоцикл".
- К символьным переменным относятся все прописные латинские буквы :
- "A", "B", "C", ..., "Z", строчные :"a", "b", "c", ..., "z", цифры :"0", "1", "2", ..., "9", знаки препинания, всевозможные скобки, русские буквы и т. д.

Символьные данные

- В программах их значения всегда заключаются в апострофы.
- Внутри каждого такого ряда коды символов упорядочены:

```
"A" < "B" < "C" <... < "Z";</li>
"a" < "b" < "c" <... < "z";</li>
"0" < "1" < "2" <... < "9";</li>
"A" < "Б" < "В" <... < "Я";</li>
```

- коды всех строчных букв меньше всех прописных.
- Для работы с таким типом данных часто используются функции, аргументы которых могут быть символьными переменными.

Функция	Назначение	Таблица 6
CHR\$ (N)	Преобразование кода N в символьное предста	авление
ASC (X\$)	Преобразование символа X\$ в десятичный к	од
LEFT\$(X\$,N)	Выделение N символов, начиная с самого лег символьном выражении X\$	вого символа в
MID\$ (X\$,N,M)	Выделение М символов, начиная с N-го симв выражении X\$ (М может быть опущено)	вола в символьном
RIGHT\$(X\$,N)	Выделение N символов, начиная с самого пра символьном выражении X\$	авого символа в
SWAP X\$,Y\$	Обмен символьными выражениями Х\$ и Ү\$	
STRINGS (N, X\$)	Формирование строки из N одинаковых симі	волов
SPACE\$(N)	Формирование строки из N пробелов	
OCT\$ (N)	Перевод десятичных чисел в восьмеричное с	числение
HEX\$(N)	Перевод десятичных чисел в шестнадцатери	чное счисление
I FN(V¢)		_

LEN(X\$)	Определение длины символьного выражения
STR\$(N)	Переводит число в символьную форму, резервируя перед символьным выражением один пробел для знака
INSTR(N,X\$,Y\$)	Поиск подстроки YS в строке X\$ начиная с N-го символа N можно опустить
·	. 17

J

2.2.6. Файл

- Одним из типов данных в QBASIC является файловый тип, который есть последовательность связанных между собой однотипных компонентов - записей, расположенных на внешнем носителе. Запись рассматривается как единое целое.
- В QBASIC имеется 2 категории файлов, работа с которыми отличается
- последовательные;
- с произвольным доступом.

2.3.Операторы языка

- Операторы языка описывают алгоритмические действия, которые необходимо выполнить для решения задачи. Сама программа представляет собой последовательность таких операторов. Каждый оператор помещается в своей строке, длина которой не превышает 255 байт. Все операторы можно разбить на выполняемые и невыполняемые.
- Выполняемые операторы служат для выполнения определенных операций или изменения порядка выполнения операторов в программе.
- К невыполняемым операторам относятся операторы управления, такие, как оператор конца программы END, объявления массивов DIM, комментария REM и т. д.
- В свою очередь, выполнимые операторы могут быть вычислительными и невычислительными.
- Кроме того, в QBASIC используются блочные операторы, состоящие из операторов.

Операторы языка

- Итак, мы рассмотрим:
- Оператор присваивания
- Операторы графики(простейшие)
- Операторы ввода- вывода
- Операторы ветвления
- Циклические операторы
- Процедуры-функции и процедуры

Оператор присваивания

$$Z = (324*x - 2 * y^3) + 525$$

Имя переменной, в которую будет занесён результат

Выражение, результат которого Вычисляется (может стоять конкретное число)

Принцип работы оператора:

- 1. Вычисляется значение выражения
- 2. При работе с арифметическими типами результат вычисления, если это необходимо, преобразуется к типу переменной
- 3. Полученный результат присваивается переменной, па прежнее её значение теряется.

Оператор **REM** – неисполняемый оператор. Вставляет в текст программы комментарий. Имеет два варианта написания: REM или '(апостроф)

Оператор вывода данных на экран:

PRINT <список вывода><завершающий символ>

Здесь: **<список вывода>** - список элементов, значения которых выводим на экран, разделённые запятыми. Элементами списка вывода могут быть как переменные, так и выражения, возможно отсутствие списка:

PRINT<завершающий символ>- в этом случае печатается пробел – курсор переводится на следующую строку.

Запятая (,) – курсор остаётся в строке вывода и очередной элемент выводится в начало очередной 14-16-символьной зоны текущей строки.

Точка с запятой (;) - курсор остаётся в строке вывода и очередной элемент выводится через 2 пробела.

Оператор ввода данных с клавиатуры:

INPUT «<текст подсказки>»;<список переменных>

Примеры: 1) INPUT «Введите Ваше имя»; М\$

2) INPUT M, P

Принцип действия оператора:

- 1. При достижении оператора исполнение программы прерывается; на экране отображаются подсказка, если она предусмотрена, и знак вопроса.
- 2. Для продолжения работы программы следует ввести с клавиатуры столько значений, сколько имён переменных указано в списке переменных, разделяя значения запятыми. Типы вводимых значений должны соответствовать типам переменных в списке.
- 3. Ввод завершается нажатием клавиши Enter. При этом переменным, указанным в списке, присваиваются набранные значения (старые значения, если они были, теряются)

 38

Оператор определения данных(констант)

DATA данные READ список переменных

Например: DATA 23, 45, 6, - 8 READ M, A, B, C



M = 23, A = 45B = 6, C = -8

Оператор DATA неисполняемый, он используется для хранения констант. В программе можно использовать несколько операторов DATA, причём они могут быть расположены в любых местах программы. Все они рассматриваются как единый «склад». А данные считываются всё равно последовательно сверху вниз и слева направо. Если данных недостаточно, то выдаётся сообщение об ошибке типа «Нет данных». Можно восстановить все данные, хранящиеся в DATA командой RESTORE

Задание оптимального графического режима экрана

SCREEN 9

Задаётся графический режим только один раз для одной задачи Размер экрана 640х350



Оператор выбора отображаемого цвета

COLOR [передний план], [фон]

где <u>передний план</u> – номер цвета отображаемых на экране текста и графики,

фон – номер цвета фона, на котором это отображение происходит.



По умолчанию, фон – чёрный, символы – белые.

Номера цветов основной палитры

Таблица цветов

Номер цвета	Название цвета	Номер цвета	Название цвета
0	Чёрный	8	Серый
1	Синий	9	Светло-синий
2	Зелёный	10	Светло-зелёный
3	Голубой	11	Светло-голубой
4	Красный	12	Светло-красный
5	Пурпурный	13	Светло-пурпурный
6	Коричневый	14	Жёлтый
7	Белый	15	Ярко-белый

42

Операторы перемещения курсора и очистки экрана

Перемещение курсора на заданную позицию экрана (строк – 25, столбцов – 80).

LOCATE <номер строки>,<номер столбца>

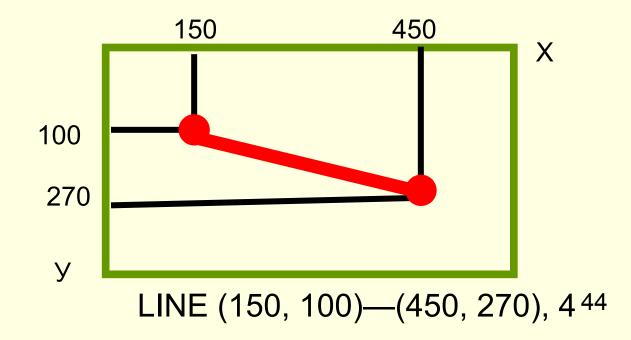
Например: LOCATE 14, 25

CLS – оператор очистки экрана

Нарисовать отрезок

LINE (X1, Y1)—(X2, Y2), C

где X1, У1 — координаты начала отрезка; X2, У2 — координаты конца отрезка; С — цвет отрезка.



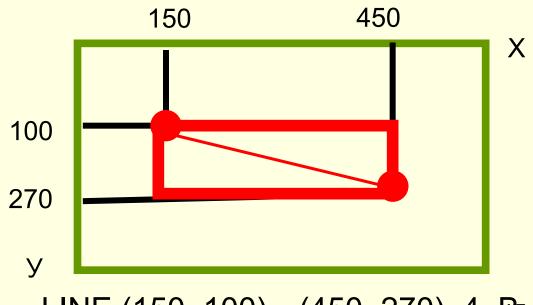
Нарисовать прямоугольник

LINE (X1, Y1)—(X2, Y2), C, B

где X1, У1 и X2, У2 — координаты начала и конца диагонали (причём, диагональ не рисуется);

С — цвет сторон прямоугольника

В – буква (признак прямоугольника от англ. слова box – ящик, коробка).



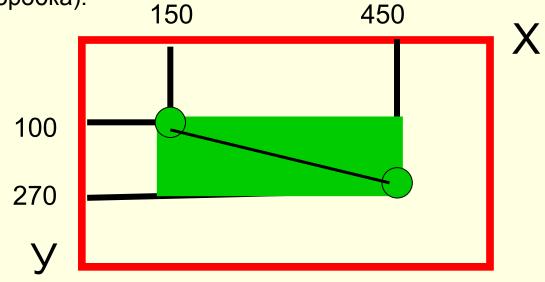
LINE (150, 100)—(450, 270), 4, **B**5

Закрашенный прямоугольник

LINE (X1, Y1)—(X2, Y2), C, BF

где X1, У1 и X2, У2 — координаты начала и конца диагонали; С — цвет

BF- признак закрашенного прямоугольника (от английского слова full box – полная коробка).

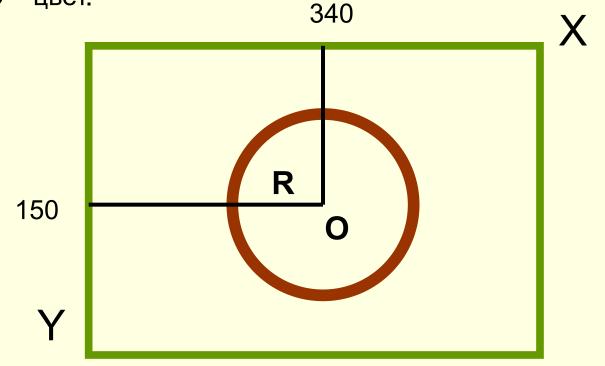


LINE (150, 100)—(450, 270), 246BF

Нарисовать окружность

CIRCLE (X, Y), R, C

где x, y – координаты центра, R – радиус (в экранных точках), C – цвет.



Например, CIRCLE (340, 150), 90, 6 47

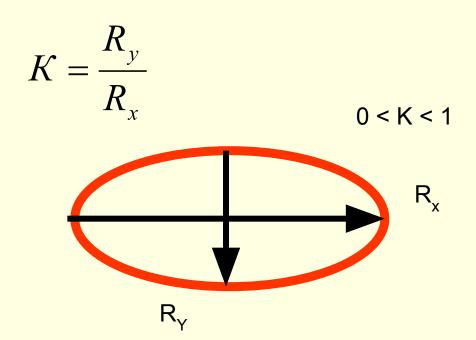
Нарисовать эллипс

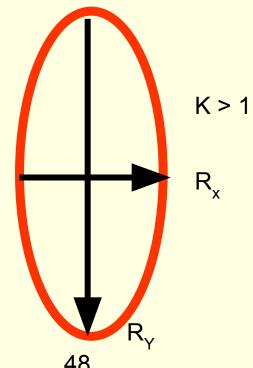
CIRCLE (X, Y), R, C,,, K

где х, у – координаты центра эллипса,

R – радиус той окружности, из которой этот эллипс получится,

С – цвет, К – значение коэффициента сжатия





Закрасить замкнутую область

PAINT (X,Y), C1, C2

где x, y – координаты любой точки внутри закрашиваемого контура, c1 – цвет, которым закрашивается область, c2 – цвет контура. Если эти цвета совпадают, то достаточно указать c1.

Правила закраски:

- контур должен быть замкнут.
- контур должен быть одноцветен. Если составляющие даже замкнутого контура разных цветов, то для компьютера эта ситуация аналогична разрыву.
- координаты точки закраски должны лежать внутри контура.

Рекомендуется закрашивать контур непосредственно после того, как он нарисован.

Если точка закраски попала:

- вне контура, то закрасится весь экран, за исключением самого контура;
- на контур, то ничего не закрасится.



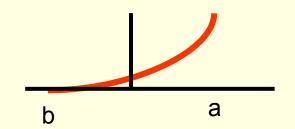
Нарисовать дуги окружности, эллипса

Оператор рисования дуги окружности

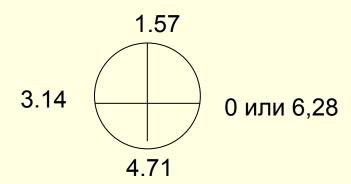
CIRCLE (X, Y), R, C, a, b

где а – угол от которого дуга начинается (в радианах)

b – угол, где дуга заканчивается



Можно указать формулу, по которой будет считаться радианная мера угла, зная величину угла в градусах (например, 30°): 3,14*30/180



Оператор рисования дуги эллипса

CIRCLE (X, Y), R, C, a, b, K

Пример

Оператор безусловного перехода

Безусловный переход оператором GOTO предписывает программе свернуть с линейного пути и перейти к метке, расположенной в любом месте программы.

GOTO метка

В качестве метки используется натуральное число. Метка указывается только в начале строки.

Данный оператор надо использовать при крайней необходимости: он портит структуру программы, делает её запутанной и затрудняет её отладку.

Оператор условного перехода

Условие — выражение, находящееся между словами «если» и словом «то» и принимающее значение «истина» или «ложь»

Условия могут быть простыми и сложными:

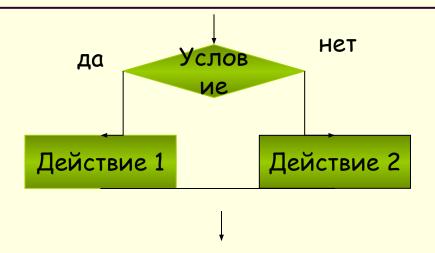
<u>Простое условие</u> — это выражение, составленное из двух переменных, арифметических выражений или двух текстовых величин, связанных одним из знаков сравнения (равно (=), не равно (<>), больше (>), меньше (<), меньше или равно (<=), больше или равно (>=)). Например, 25 > 3, X + Y <= 45

Сложное высказывание - это последовательность простых условий, объединённых между собой знаками логических операций (и (and), или (or), не (not)).

Например, (17 > X) and (X < 19)

Ветвление

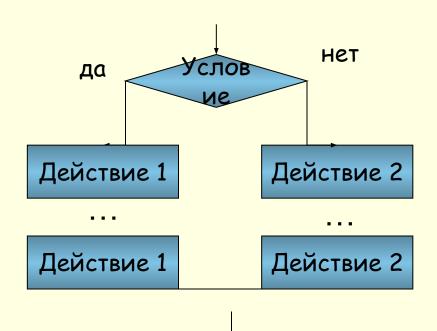
IF <условие> THEN <действие 1> ELSE <действи2>



Принцип действия оператора:

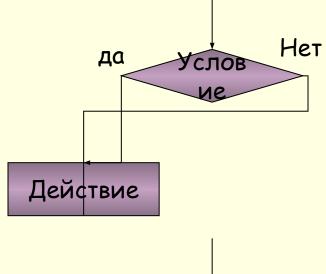
- 1. Вычисляется выражение при текущих значениях входящих в него переменных.
- 2. Если полученное значение есть истина, то выполняется <действие1> (ветвь THEN), а <действие2> игнорируется. Если полученное значение ложно, то выполняется <действие2> (ветвь ELSE), а <действие1> игнорируется.
- 3. После выполнения одной из ветвей управление передаётся следующей строке программы.
- 4. Весь оператор должен располагаться в одной строке. 53

Если в каждой из ветвей IF-конструкции выполняется не один оператор, а несколько, то их можно разделить двоеточием. А ещё лучше записать конструкцию в другом виде (теперь запись оператора в одну строчку не обязательна:



Неполное ветвление

Если ветка ELSE отсутствует и условие ложно, то управление сразу передаётся следующей строке программы: никакие из операторов не выполняются.



Оператор:

IF <условие> THEN <действие >

Самостоятельно

<u>Задача 1.</u> Напишите программы, которые в зависимости от введенного числа либо вычисляют функцию, либо выдают сообщение, что функция не определена:

a)
$$y = \frac{1}{x}$$
 6) $y = \sqrt{x^2 - 1}$

Задача 2. Напишите программу для вычисления функции:

$$y = \begin{cases} \frac{\cos x}{x}, & x > 0\\ x \sin x, & x \le 0 \end{cases}$$

Задача 3. Напишите программу, определяющую четность или нечетность введенного с клавиатуры целого числа.

Задача 4. Напишите программу, находящую меньшее из двух, введенных с клавиатуры чисел.

Оператор выбора SELECT

Конструкция оператора:

```
SELECT CASE <тест выражение>
                                     <тест выражение> - любое числовое или строковое
                                     выражение;
  CASE <список выражений1>
                                     <писок выражений1> - одно или несколько
  [блок операторов1]
                                     выражений для сравнения с параметром <тест
   CASE <список выражений2>
                                     выражение>
  [блок операторов2]
                                     [блок операторов2] – один или несколько операторов
                                     в одной или нескольких строках
                                     Аргументы списка выражений могут принимать любую
 CASE ELSE
                                     форму:
   [блок операторов n]
                                    -Выражение [,выражение]...
END SELECT
                                    -Выражение ТО выражение
                                    -IS оператор отношения выражения
                     INPUT «Введите номер месяца»; N
           Пример:
                SELECT CASE N
                   CASE IS >=12
                   PRINT «не существует такого месяца»
                   CASE IS <= 0
```

PRINT «не существует такого месяца»

CASE 1 TO 12

CASE 1

END SELECT

PRINT «такой месяц есть»

PRINT «понедельник»

57

Оператор цикла с заранее известным числом повторений FOR...NEXT

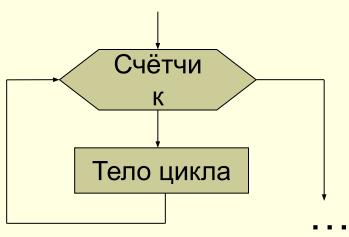
FOR <переменная> = <выр1> TO <выр2> [STEP <выр3>] <тело цикла>

NEXT <переменная>

<переменная> - управляющая переменная (параметр, счётчик) цикла. Может быть любая переменная арифметического типа.

<выр1>,<выр2>,<выр3>- арифметические выражения, определяющие соответственно начальное значение параметра, конечное его значение и приращение (шаг). Шаг может быть положительным и отрицательным. Если шаг равен 1, то заключённую в квадратные скобки часть оператора можно опустить.

<тело цикла> - любое количество строк, содержащие операторы, которые требуется повторять.



Принцип действия оператора:

- 1. Вычисляются выражения <выр1>,<выр2>,<выр3>
- 2. Параметру присваивается начальное значение <выр1>
- 3. Значение параметра сравнивается с конечным значением <выр2>
- Если значение параметра не больше <выр2> (при положительном шаге) или не меньше <выр2> (при отрицательном шаге), то выполняется тело цикла
- В противном случае осуществляется выход из цикла, и выполняется оператор, следующий за ключевым словом NEXT
- 4. Если не произошёл выход из цикла, значение параметра изменяется на величину шага, и повторяется п.3

Пример записи оператора с параметром:

Определить результат выполнения программы:

Решение:
$$K = 1$$
, $(1 < 8)$ $X = 0 + 1 = 1$ $K = 3$, $(3 < 8)$ $X = 1 + 3 = 4$ $K = 5$, $(5 < 8)$ $X = 4 + 5 = 9$ $K = 7$, $(7 < 8)$ $X = 9 + 7 = 16$ $K = 9$, $(9 > 8)$ выход

Ответ X=16

ЗАДАНИЕ:

Определите, какие из представленных операторов написаны правильно, а какие нет.

Ответы:

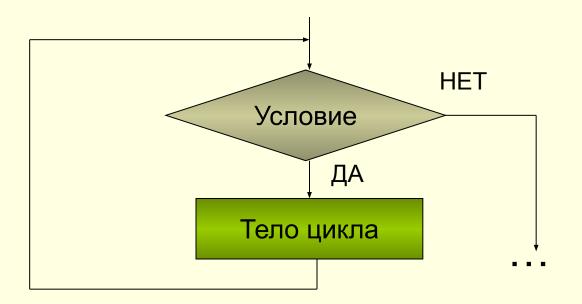
- 1. FOR D = R TO S STEP H
- 2. FOR S = 2 6
- 3. FOR S = 4 TO 12 STEP 2
- 4. FOR S 12 TO 4 STEP 2

- 1. Правильно, если переменные R,S,H ранее получили значения
- 2. Неправильно, отсутствует ключевое слово TO
- 3. Правильно
- 4. Неправильно, цикл выполняться не будет, т.к. начальное значение параметра больше конечного, а шаг неотрицателен.

Оператор с неизвестным числом повторений DO...LOOP и WHILE...WEND

Итерационные циклы делятся на 2 типа: цикл с предусловием и постусловием.

В циклах с предусловием осуществляется проверка условия цикла, а потом, если оно истинно, выполняется тело цикла.



Операторы цикла с предусловием

Оператор 1

Оператор 2

WHILE <выражение> <тело цикла> WEND

DO UNTIL <условие> <тело цикла> LOOP

Оператор 3

DO WHILE <условие> <тело цикла> LOOP

Если используется ключевое слово **UNTIL**, цикл будет выполняться только тогда, когда условие имеет значение «ложь», если **WHILE** цикл будет выполняться только тогда, когда условие имеет значение «истина».

Примеры использования цикла с предусловием

1) WHILE S < 10 PRINT S WEND

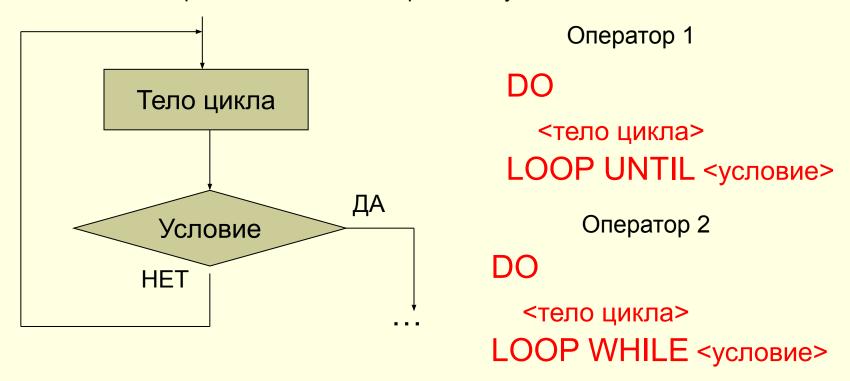
Пояснение: пока S < 10, то значение S печатаем

2) WHILE WORD\$ <> "CURS"
INPUT «Введите пароль»; WORD\$
WEND
PRINT «Правильно! Вы допущены к тесту »

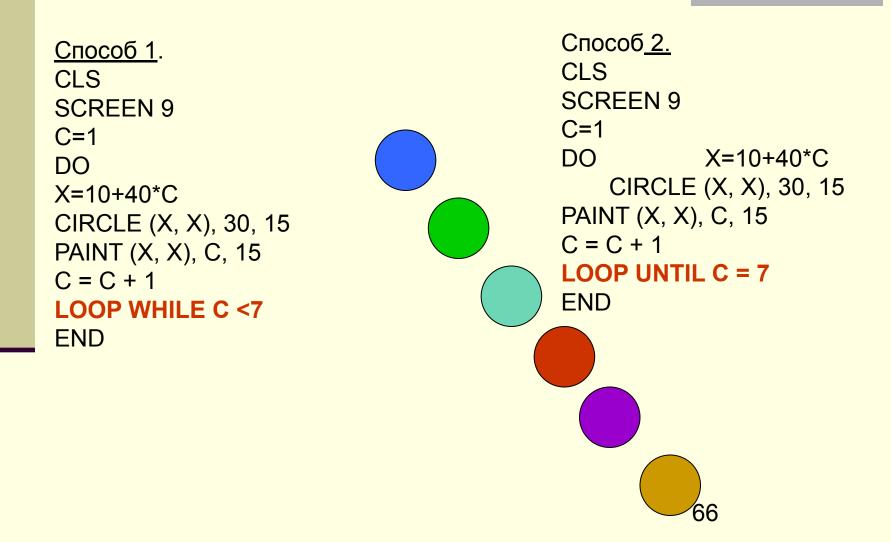
Пояснение: пока введённый пароль не совпадёт со словом CURS, выхода из цикла нет

Операторы цикла с постусловием

В циклах с постусловием выполнение тела цикла предваряет проверку условия, поэтому он непременно выполняется хотя бы один раз. Это делает его источником ошибок при невнимательном использовании; кроме того, его использование действительно оправдано в достаточно редких случаях.



ПРИМЕР: Составить программу, которая выводит на экран круги, расположенные по диагонали и закрашенные цветами с 1 по 6.



Цикл с несколькими параметрами

Бывает, что при попытке выполнить задание с помощью оператора цикла выясняется, что изменяется не одна, а две или три величины. В этом случае, если вы можете установить между ними зависимость, следует использовать следующие правила:

- 1. определить, какие же величины изменяются, и обозначить их буквами
- 2. назначить одну из этих величин независимым аргументом
- 3. определить закономерности, связующие изменяющиеся величины, и выразить эти величины через независимый аргумент
- 4. написать программу с оператором цикла, основным параметром которого будет независимый аргумент, а в командах тела цикла, на месте других изменяющихся величин, необходимо подставить их выражения через этот независимый аргумент.

Вложенные циклы

В предыдущем разделе мы рассматривали случаи, когда в цикле изменяются несколько величин, но их можно выразить друг через друга. Сейчас мы рассмотрим ситуацию, когда изменяются несколько независимых величин.

FOR <переменная1> = <выр1> **TO** <выр2> [**STEP** <выр3>]

FOR <переменная2> = <выр4> **TO** <выр5> [**STEP** <выр6>]

<тело цикла>

NEXT < переменная 2>

NEXT < переменная 1>

Оператор приостановки выполнения программы и оператор звука

Оператором **SLEEP** приостанавливает выполнение программы на указанное количество секунд, например

SLEEP 10 (останавливает на 10 сек.)

Если значение секунд указано 0 или не указано вовсе, то программа будет ожидать нажатия любой клавиши.

Оператор **BEEP** подаёт на встроенный динамик звук с частотой 800Гц и длительностью 1,4 секунды.

Случайные числа

Для создания какой-либо последовательности случайных чисел служат показания встроенного в компьютер таймера. Чтобы инициировать процесс генерации последовательности случайных чисел используется оператор

RANDOMIZE TIMER

Затем, чтобы получить из этой последовательности какое-либо значение, мы должны использовать оператор RND (N), который выдаёт случайное число от 0 до 1. Минимальное полученное число 0,0000001, а максимальное 0,9999999. В качестве N может служить любое целое или действительное число. Этот параметр также влияет на выбор компьютером случайных чисел.

```
Пример: RANDOMIZE TIMER

X = RND(1)

PRINT X

END
```

Программа «Звёздное небо»

Изобразим звёздное небо 500 белыми точками на чёрном фоне. Координаты каждой точки должны быть в пределах по X от 0 до 640 и по У от 0 до 350.

```
CLS
SCREEN 9
RANDOMIZE TIMER
FOR N = 1 TO 500
X= INT (RND (1) * 641)
Y = INT (RND (1) * 351)
PSET (X,Y), 15
NEXT N
END
```

Построение графиков функции

Графики строятся по точками, а функция для построения задаётся компьютеру оператором

DEF FN имя_функции (параметр_функции) = функция

Например, DEF FNY(X) = SIN(X)

Кроме того, координатная сетка задаётся следующими операторами

Программа построения графика Y(X) = SIN(X)

```
CLS

SCREEN 9

LINE (0, 175) — (640, 175), 15

LINE (320, 0) — (320, 350), 15

DEF FNY(X) = SIN(X)

FOR X = 0 TO 640

Y = 175 - 30 * FNY ((X - 320) / 30)

PSET (X, Y), 15

NEXT X

END
```

Достаточно сложное выражение для У связано с переносом начала координат в центр экрана, число 30 в данном случае произвольно – это коэффициент растяжения

Символьные (или строковые) переменные

— переменные, значениями которых являются строки символов, заключённых в кавычки.

ASC(X\$) - функция, переводит первый символ строки в ASCII-код Пример: ASC("A") = 65,
ASC("BIT") = 66

CHR\$(X) - функция, переводит ASCII-код X в символ Пример: CHR\$(67) = "C"
CHR\$(68) = "D"

OCT\$(X) — функция, переводит число в восьмеричную запись цифр (не в число!).

Пример: b = 10OCT\$(b) = "12" OCT\$(25) = "31"

INKEY\$ - возвращает символ, считанный с клавиатуры Операция конкатенации строк: "ВО" + "ДА" - 4" ВОДА" INSTR(N, X\$, Y\$) – функция, выполняет в строке X\$ поиск подстроки Y\$ с позиции N(необязательный параметр).

```
Пример: a$ = "клавиатура", b$ = "тур"

INSTR (a$,b$) = 7

INSTR (4, a$, "a") = 6
```

LCASE\$(X\$) – преобразует все буквы строки X\$ в строчные Пример: а\$ = "Бим - Бом" LCASE\$(a\$) = "бим - бом"

UCASE\$(X\$) – преобразует все буквы строки X\$ в прописные Пример: а\$ = "Бим - Бом" UCASE\$(a\$) = "БИМ - БОМ"

LEFT\$(X\$, N) – функция, выделяет N левых символов строки X\$ Пример: a\$ = "паровоз" LEFT\$(a\$,3) = "пар" LEFT\$("сокол", 3) = "сок"

RIGHT\$(X\$, N) — функция, выделяет N правых символов строки X\$ Пример: a\$ = "паровоз" LEFT\$(a\$,3) = "воз" LEFT\$("вода", 2) = "да"

LEN(X\$) – функция, возвращает длину строки X\$

Пример: a\$ = "дискета" LEN(a\$) = 7 LEN("экран") = 5

MID\$(X\$, Y, Z) – функция, выделяет часть строки X\$ из Z символов с позиции Y, если Z не указано, то выделяются все оставшиеся символы.

Пример: a\$ = "сокол"

MID\$(a\$, 2, 3) = "око"

MID\$("сокол", 3) = "кол"

VAL(X\$) – функция, переводит строку X\$ в десятичное число. Символьная переменная должна иметь правильное представление десятичного числа.

Пример: a\$ = "15 bit" VAL(a\$) = 15 VAL("ABC 5") = 0

MID\$(X\$, Y,Z,)=F\$ - оператор, который заменяет Z символов с позиции Y в строке X\$ символами строки F\$.

```
Пример: a$ = "класс"

MID$(a$,2, 3) = "оло"

A$ = "колос"
```

SPACE\$(X) – функция, возвращает строку пробелов. Пример: SPACE\$(5) = " (пять пробелов)

STR\$(X) – функция переводит число в строку.
Пример: b = 10
STR\$(b) = "10" (с лидирующим пробелом)

STRING\$(N, K или X\$) - функция возвращает строку длины N из символов с кодом K или первым символом заданной строки X\$ Пример: STRING\$(3, A) = "AAAA" STRING\$(5, 65) = "AAAAA

Однобайтовая кодировка символов кодами ASCII (American Standard Code for Information Interchange – Американский стандартный код обмена информацией)

позволяет кодировать 256 различных символов и команд. Наиболее часто требуемые коды при решении задач:

- От 48 до 57 коды цифр от 0 до 9
- От 65 до 90 прописные буквы латиницы (A 65, B 66 и т.д.)
- От 97 до 122 строчные буквы латиницы (а 97, b 98 и т.д.)
- От 128 до 159 прописные буквы кириллицы (A 128, Б 129 и т.д.)
- От 160 до 175 строчные буквы кириллицы
- Клавиши Таb 9, Esc —27, пробел 32, точка 46, запятая
 44, вопросительный знак 63, восклицательный знак 33 и

т.д.

Массивы

Массив – это набор однотипных данных (чисел, символов, слов), которые хранятся в одном месте памяти компьютера в упорядоченных по номерам ячейках.

Массив имеет:

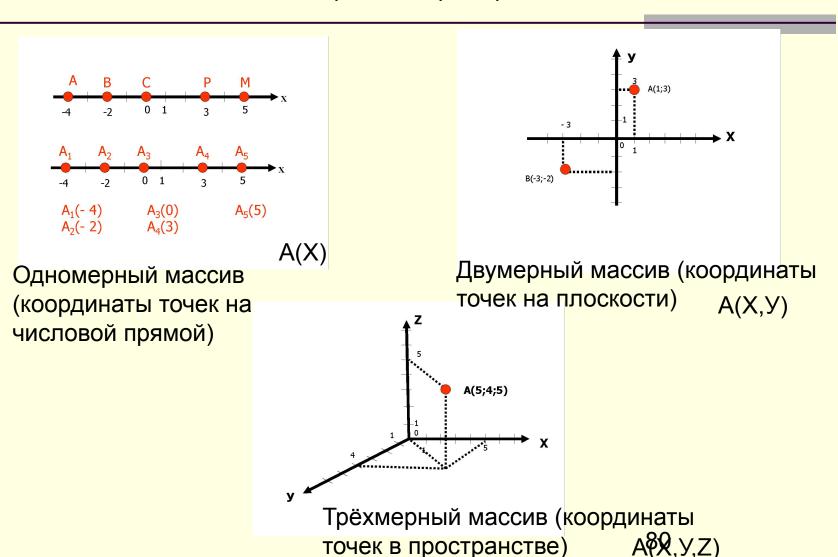
- Имя
 - Тип (%- целочисленный, \$ символьный, без знака вещественный)
 - Размерность количество элементов
 - Индекс номер элемента

Каждый элемент массива в общем виде описывается как B(i), где B - имя массива

i – номер или индекс элемента массива

Если адрес элемента массива определяется одним индексом, то такой массив называется одномерным.

Массивы бывают и многомерными. Примеры:



Перед тем как начать работать с массивом, нужно зарезервировать для него место в памяти. Для этого имеется оператор DIM (от английского слова dimention –объём, величина, размер, измерить)

Пример. **DIM A(10)**

Обычно говорят, что зарезервировано место под 10 элементов массива A, однако в самом деле резервируется 11, т.к. нумерация ячеек начинается с 0. Но так как нам привычнее начинать считать с 1, то первая («нулевая») ячейка просто-напросто не используется.

Способы заполнения одномерных массивов

C клавиатуры INPUT "N =", N DIM M(N) FOR I = 1 TO N INPUT M(I) NEXT I

Заранее известными значениями

DATA 23, -13, 9.8, 77, 45 DIM M(5) FOR I = 1 TO 5 READ M(I) NEXT I

При помощи стандартных функций

RANDOMIZE TIMER
INPUT «N=»; N
DIM M(N)
FOR I = 1 TO N
M(I) = INT(RND(1)*100)
NEXT |

Непосредственное присваивание значений элементам

DIM D(3) D(1)=12.6 D(2)=5.96 D(3)=98





Процедуры – функции и процедуры

Процедурой называется часть программы, реализующая вспомогательный алгоритм и допускающая многократное обращение к ней из различных мест основной программы.

процедура-функция FUNCTION...END FUNCTION

Синтаксис: FUNCTION имя [список][STATIC]

[операторы]

имя = выражение

[операторы]

END FUNCTION

Параметры:

- -имя объявляет имя функции
- **-список** это список разделённых запятой аргументов формальных параметров, которым из основной программы передаются и присваиваются значения аргументов фактических параметров
- атрибут **STATIC** указывает, что переменные являются локальными в функции и сохраняются между её вызовами.
- -Запись **имя = выражение** возвращает значение функции, присвоенное её имени

Процедуры

Оператор SUB...END SUB выделяет начало и конец процедуры.

Синтаксис: SUB имя [(список)] [STATIC]

[операторы]

[EXIT SUB]

операторы

END SUB

- имя глобальное имя процедуры, ограниченное длиной в 40 символов
- **СПИСОК** СПИСОК, разделённых запятыми имён переменных, передаваемых процедуре при её вызове.

В операторе предусмотрен альтернативный выход с помощью **EXIT SUB.** В отличие от процедуры-функции FUNCTION имя процедуры SUB не может быть использовано в выражениях. Процедуры могут быть *рекурсивными*, т.е. могут вызывать сами себя.

Вызов процедуры SUB...END SUB выполняется оператором CALL.

Синтаксис 1: CALL имя процедуры [(список аргументов)]

Синтаксис 2:имя процедуры [список аргументов]

83

КОНЕЦ



ПЛАН части 2

- Оператор присваивания
- Операторы ввода вывода
- Графика в QBasic
- Операторы ветвления
- Циклические операторы
- Строковые операторы и функции
- Массивы
- Процедуры-функции и процедуры