Первый учебник - глава 1

Глава ПЕРВАЯ. Добро пожаловать! На этих уроках вы научитесь, как использовать
язык программирования QBASIC. Если вы хотите изучить Бейсик, то, преступим!

Вы можете спросить, каков он, QBASIC. QBASIC - язык программирования, написанный на компьютерах
в 1975, Биллом Гейтсом и Паулем Алленом, и с тех пор был приятен всем. Почему?
Из-за легкости в использовании, команды - чистый английский язык, и мощность. QBASIC заменяет Универсальную Символическую систему команд. Несколько команд - чистый Английский язык - PRINT, LET…, и многие другие. Язык имеет простую структуру программ: строки пронумерованы (10, 20, 30, и т.д.). Но почему вы решили использовать QBASIC? Когда вы изучаете QBASIC,
вы также изучаете многие из основных принципов других языков программирования.
Изучив учебник, вы сами сможете создавать программы, и, я думаю, найдёте в этом забаву:))).
К концу этих уроков, вы должны знать много вещей относительно QBASIC, и будете способны писать превосходные программы, которые будут являться полезными для вас и ваших друзей:))

QBASIC очень лёгок в использовании. Почти на каждом компьютере записан QBASIC.
Если Вы ещё не знаете, где QBASIC записан на вашем компьютере, проверьте каталог "C:\DOS",
или используя программу поиска, найдите QBASIC.EXE.

Я знаю, что вы хотите сразу же, научится программированию и написать первую программу.
Есть программа для вас. Напишите в окне Бейсика:

```
CLS:PRINT "Привет"! : PRINT : PRINT " Это моя первая программа! "
```

Это только список команд, которые компьютер будет выполнять.
Нажмите на меню "RUN",> "START". На экране появится что-то вроде:

```
Привет!
Это - моя первая программа!
```

Обратите внимание, что QBASIC не выводит CLS, или PRINT, или что-нибудь другое.
Нажмите любую клавишу, чтобы вернуться к программе. Объяснение операторов:

CLS - Очищает экран

Оператор PRINT - печатает написанный вами в кавычках текст.

Чтобы сохранить программу нажмите на меню "FILE">Save as…, наберите имя файла
укажите директорию и сохраните файл. Чтобы начать новую программу,
нажмите на меню "FILE">"NEW" ОСТЕРЕГАЙТЕСЬ - любая программа в памяти будет потеряна НАВСЕГДА, если она не сохранена. Если Вы хотите загрузить свою программу снова, нажмите на меню "FILE",
нажмите "LOAD", зайдите в ту директорию где лежит ваша программа выберете её и загрузите.

Итак в этой главе мы изучили 2 оператора "PRINT" и "CLS".

Глава 2

Глава ВТОРАЯ. Добро пожаловать! В этой главе, мы изучим следующие команды: LET, INPUT.
Давайте, начинать!

В предыдущей главе, Вы научились использовать CLS и PRINT.
В этой главе, вы узнаете о переменных и команде INPUT.

Что такое переменные? Переменные - "поля" в памяти компьютера, для сохранения значений,

номеров, названий, чисел. Имеются два основных типа переменных - числа и "строки",
которые являются текстовыми переменными. Переменным дают их тип символом после их названия.
Категория "чисел" далее разделена на четыре области. Нормальный шрифт,
называемый целыми числами, не требует никакого символа, или может сопровождаться %.
Они могут быть в диапазоне от -32767 до 32767. Целые числа - то,
что Вы будите использовать большинство времени.

Другой тип переменных, длинные целые числа, имеют диапазон -2 миллиардов до 2 миллиардов.
Вы можете спросить, почему не делают все числа длинными целыми числами?. Есть ответ.
Память в компьютере, особенно в QBASIC, ограничена, и вы должны занять так мало пространства,
насколько возможно. Вы должны использовать длинные целые числа только там, где они необходимы.

Четвертый тип чисел - числа "с плавающей точкой". Эти числа - десятичные переменные,
которые могут иметь очень длинные десятичные пространства.
Короткие типы (отмечаются - !) и длинные типы (отмечаются - #).
Эти переменные обычно не используются, если вы не делаете определенные функции учета.
Назначают переменные используя команду LET. Например:

```
LET number = 123
```

Это присваивало значение 123 переменной "number". Вы также можете использовать математические функции при назначении переменных:

```
LET number = 4 * 12
```

Это присвоит переменной "number" цифру 48. Вы можете увеличивать переменные подобно этому:

```
LET number = number + 1
```

Это значит, что номер переменной "number" становится больше на 1. Вы можете также прибавлять две переменной вместе. Теперь Вы знаете, как назначить значение на переменную, используя команду LET.

Вы хотите вывести эти переменные на экран. Используете команду PRINT для этой задачи:

```
PRINT number
```

Это выведет на экран значение переменной "number".
Если вы хотите перед переменной напечатать текст, вы должны написать так:

```
LET number = 100: PRINT " Номер равен "; number
```

Это выведет на экран при запуске:

```
Номер равен 100
```

Теперь вы спросите, как самому вводить переменные. Для этого используют команду INPUT.
Например:

```
INPUT "Как вас зовут?"; username$ : PRINT "Привет, "; username$; "."
```

При запуске программы компьютер спросит как вас зовут, после того как вы наберете своё имя,
он его напечатает. В этой главе, мы изучили команду PRINT и узнали операторы LET и INPUT.

Задание на эту главу :

Напишите программу, при запуске которой вам предлагалось бы ввести ваше имя и возраст,
и после того, как вы ввели имя и возраст программа должна была бы вывести имя и возраст на экран,
используя команду PRINT.

Глава 3

ТРЕТЬЯ Глава. Привет! В этой главе, мы изучим следующие команды:

DIM, FOR...NEXT, STEP, GOTO, IF...THEN, COLOR

Сейчас, вы знаете: Относительно типов переменной, операторы PRINT, INPUT, и LET.
Это базовые детали из среды БЕЙСИКА, но теперь мы должны двигаться в комплексную область
БЕЙСИКА.

Переменные используются в каждой программе, независимо от того какая программа это.
Именно поэтому переменные настолько важны в БЕЙСИКЕ. Однако, когда много строк,
сотни переменных, названия которых трудно запомнить, вводят МАССИВЫ.

Массивы - большое поле в памяти компьютера. Вы можете помещать различные вещи в поля массивов,
но их организовывают в одно большое поле. Чтобы создавать массив, используют команду DIM:

DIM ArrayName$ (number)

ArrayName - название массива, сопровождаемого типом переменной($,!,# и т.д.)
номер в круглых скобках - количество "полей" в массиве. Чтобы использовать массив, назначают
значения массива, используя команду LET. Например:

```
DIM Day$(2)
LET Day$(1) = "Воскресенье
LET Day$(2) = "Понедельник
```

И так далее. Чтобы ПЕЧАТАТЬ одну из этих переменных в массиве,
пишите точно так же как и для печати переменной (т.е. с помощью команды PRINT):

```
PRINT "Сегодня: "; Day$(1); "."
```

И, соответственно, чтобы получить значение от клавиатуры в массив, используете команду INPUT:

```
INPUT "Как зовут первого игрока?"; players$(1)
INPUT "Как зовут второго игрока?"; players$(2)
```

Вы можете использовать элемент массива также, как переменную. При использовании массивов, вы
можете управлять сотнями названий, значений: Строки, долларовые количества, и намного больше.
Теперь вы умеете использовать массивы. Вы, я думаю, не смогли бы, скажем,
напечатать 100 различных значений массива, названного "players$".
Это очевидно было неэкономично, иметь 100 различных команд печати.

Поэтому, нам нужно изучить ЦИКЛЫ. Цикл - набор команд, которые повторены несколько раз,
до какого либо условия, или бесконечно. Первый цикл, который мы изучим команда FOR...NEXT.

Это повторяет команды между операторами FOR и NEXT столько раз, сколько указанно.
Синтаксис прост:

```
FOR i = 1 TO 100
PRINT i
NEXT i
```

Программа печатала бы числа от одного до сотни. Вместо переменной "i", можно давать любое имя, например, "bob" или "sam" или "linda", или "z".
Вы можете так же брать более низкие или высокие значения переменной - например:

```
FOR i = 6 TO 371
PRINT "Магический номер может быть:"; i
NEXT i
```

Другой способ применения FOR...NEXT - способ приращения цикла - использованием команды STEP.
Если вы опускаете команду STEP (как в обычном FOR...NEXT цикле),
"i" увеличивается каждый раз после выполнения цикла. С использованием команды STEP,
Вы можете изменять её на любой номеру какой пожелаете. Например:

```
FOR i = 100 TO 300 STEP 2
PRINT "Магический номер может быть:"; i
NEXT i
```

Это бы вывело все четные числа от 100 до 300.
Вы могли бы также заставить переменную "i" отчисляться назад, как в этом примере:

```
FOR i = 300 TO 100 STEP -2
PRINT "цифра - теперь:"; i
NEXT i
```

FOR...NEXT циклы могут использоваться везде. В следующей главе,
мы изучим два других видах цикла, которые продолжают выполнятся до некоторого условия.
В этой части мы изучим оператор GOTO.
Если Вы не знаете оператор GOTO, то заверяю вас, оператор чрезвычайно прост.
GOTO - мощная команда, которая позволяет вам переходить в разные части программы по номерам строки. Но для этого вы должны "маркировать" строки, к которым вы хотите перейти.
Например:

```
topofprogram: CLS
PRINT "Это бесконечный цикл"
GOTO topofprogram
```

Или, так:

```
1 CLS
PRINT "Привет мир!"
GOTO 1
```

GOTO - очень простой, но все же мощный оператор.
Но использование оператора не обязательно в вашей программе, его можно заменить другим.
GOTO становится очень полезным, когда используется вместе с операторами IF...THEN.

Вы можете использовать GOTO, чтобы создать некоторый тип меню, подобно этому:

```
PRINT "Мое Меню"
PRINT "Нажмите 1, чтобы очистить экран, и 2, чтобы вывести "Привет"!"
INPUT "Что вы выбираете"; choice
IF choice = 1 THEN GOTO clrscr
IF choice = 2 THEN GOTO hello
clrscr: CLS
PRINT "Готово."
hello: PRINT "Привет, Привет, Привет!"
END
```

Вы можете использовать GOTO после THEN, чтобы перейти к другой части программы программы.
Или, Вы можете заменять "=" любым математическим символом
(подобно знаку больше ">" или меньше "<").
Есть пара примеров в одной программе:

```
PRINT "Пример Программы #1"
PRINT "Номер, о котором я думаю стоит в промежутке от 1 и 10."
PRINT "Вы получаете 3 возможности."
INPUT "Первый выбор"; number
IF number = 3 THEN GOTO gotit
PRINT "Простите!"
INPUT "Второй выбор"; number
IF number = 3 THEN GOTO gotit
PRINT "Нет!"
INPUT "Последний выбор"; number
IF number = 3 THEN GOTO gotit
PRINT "Простите! The number was 3!"
END
gotit: PRINT "Вы выиграли! Хорошая работа!"
```

Основной элемент в этой программе - IF...THEN.

Последняя вещь в этой главе – изучение изменения цвета текста и других элементов.
Цвет текста на экране позволяет менять команда COLOR (Включая Цвет фона).
Вот пример использования этого оператора:

```
COLOR 13
PRINT "Сиреневый!"
COLOR 7
PRINT "Серый!"
```

Число, после оператора COLOR - один из этих номеров цвета:

00 - Сажа
08 - Тёмно - серый
01 – Темно - синий
09 – Светло - синий
02 – Тёмно – зеленый
10 - Светло - зеленый
03 – Тёмно - голубой

11 – Светло - голубой
04 – Темно - красный
12 - Светло - красный
05 – Темно - пурпурный
13 - Сиреневый
06 - Цитрус
14 - Жёлтый
07 - Серый
15 – Ярко - белый

Это заканчивает наше обсуждение оператора COLOR и нашей главы.
В этой главе узнали много нового.

Глава 4

Глава Четвёртая. Привет! В этой главе, мы изучим следующие команды:

DO...LOOP, OPEN, INT, CLOSE, RANDOMIZE TIMER RND, PRINT#, SELECT...END, SELECT, INPUT # INKEY$

В предыдущей главе, мы говорили относительно циклов DO...LOOP, которые позволяют повторять действие некоторое количество раз. Окончание цикла при уловии:
Все, что вы должны сделать - написать, DO WHILE... или UNTIL..[какое-либо условие].
В переводе с английского это бы звучало так:
ДЕЛАЙТЕ какое - либо ДЕЙТСТИЕ ДО ТОГО КАК ПРОИЗОЙДЁТ [какое-либо условие]
Обязательна команда LOOP, чтобы закончить цикл.
Есть очень простой пример из того, как использовать этот цикл:

```
DO UNTIL a = 10
try = try + 1
PRINT "Попробуйте номер"; try
PRINT
PRINT "Наберите секретный номер!"
INPUT "Введите"; a
LOOP
```

Это бы значило что цикл продолжался до набора числа 10. Эта команда очень простая,
но бесконечно полезная.
Отличие этого цикла от команды INPUT - не нужно нажимать клавишу "ENTER" после ввода числа.
Сейчас программисты стремятся придерживаться этого в своих программах,
а именно, в местах где написано "нажмите любую клавишу".
Функция INKEY$ - требует немедленного нажатия на любую клавишу…
Вы можете "вставлять" эти команды в вашу программу.

```
DO WHILE INKEY$ = ""
PRINT "Нажмите любую клавишу. Для окончания цикла..."
LOOP
```

И это бы значило, то что, цикл бы не прекратился пока бы вы не нажали любую клавишу.
Функцию можно использовать в:
Меню "вызывающей клавиши" INPUT(Как горючую клавишу).
Однако, это требует конструкцию, подобную IF...THEN.

Изучим, теперь, новый набор команд - SELECT CASE...END SELECT,
что позволяет вам устанавливать связку IF...THEN,
вместо сотен отдельных функций в вашей программе будут операторы SELECT CASE...END SELECT.
Это также позволит вам иметь многократные команды в переменной CASE.
Вот простой пример:

```
PRINT " Главное меню "
PRINT "1) Окончание программы"
PRINT "2) Сюрприз"
PRINT
INPUT "Ваш выбор"; chc
SELECT CASE chc
CASE 1
PRINT "Хорошо неправда ли!"
END
CASE 2
PRINT "СЮРПРИЗ!!!!!!!!"
PRINT "Вы были удивленны!!! А?"
END
CASE ELSE 'обратите сюда внимание
PRINT "Почему вы выбрали 1 а не 2?" 'команда, которая позволяет вам  END
END SELECT' захватывать недопустимые ответы (CASE ELSE)
```

Все, что вы должны сделать для работы программы - выбрать один из 2-ух разделов переменной "chc".
Когда вы выберите раздел, ваша программа будет выполнятся.
Если же вы не выбираете, то программа стоит на месте. Интересное место программы там,
где я использовал оператор CASE ELSE. Например, если вы не выбрали 1 то этот оператор позволяет
выполнять программу под остальными цифрами т.е. в нашем случае под цифрой 2.
Выбор должен обязательно начинаться с оператора CASE SELCET,
а заканчиваться оператором END SELECT.
Изучим следующий полезный пункт - как делать случайные числа в QBASIC.
Очень простой способ сделать это, можно сделать с помощью команд LET, INT, и RND.

Имеется простой пример этого:

```
x = INT (RND * 10) + 1
```

Это бы присвоило переменной "x" между число 1 и 10. Получить номер
Между 0 и 10, а не 1 и 10 можно избавившись " + 1 " в конце строки.
Это все, что вы должны знать, чтобы сделать генерацию случайных чисел:

INT - округляет дробное число до целого.
RND - задаёт случайные числа

Теперь, перейдём к командам файла. Это обязательно должно быть изучено!
Для открытия файлов используется команда OPEN, чтобы закрыть файл команда CLOSE.
Файл должен обязательно закрываться, иначе ваша программа не будет работать!
Чтобы открыть файл, используют команду OPEN таким образом:

```
OPEN "filename.ext" FOR (OUTPUT/INPUT/APPEND) AS #1(or other number)
Открыть "Имяфайла.расширение" для (Ввода/Вывода/Присоединения) Как  #1(или другой номер)
```

Команда OPEN гораздо более сложна чем показано на этом примере,
но я попытаюсь всё это объяснить. Есть несколько типов открытия файлов:[OUTPUT/INPUT/APPEND]
для Ввода, Вывода и Присоединения. Если вы хотите прочитать из файла,
строку или несколько строк, то используйте оператор INPUT. Записывать в файл,
вы должны использовать команду OUTPUT. И присоединять к файлу (прибавлять к концу файла),
вы должны использовать команду APPEND. Разберём как всё это работает.
Допустим, мы открыли наш файл. Теперь, что мы делаем?
Вы все еще помните команды PRINT и INPUT, так в зависимости от того,
что вы делаете вводите ли вы или выводите. Используют команду PRINT,
чтобы печатать и команда INPUT, чтобы читать из файла.
Нужно так же задавать номер файла, (например#1).
Есть пример:

```
OPEN "file.txt" FOR OUTPUT AS #1
PRINT #1, "Привет мир!"
CLOSE

OPEN "file.txt" FOR INPUT AS #1
INPUT #1, s$
PRINT s$
CLOSE
```

Вы должны использовать LINE INPUT, чтобы читать полную строку в файле.
Например:

```
OPEN "file.txt" FOR INPUT AS #1
LINE INPUT #1, s$
PRINT s$
CLOSE
```

Это читало бы полную строку из файла в переменную "s$". Только заново
элемент, при использовании OUTPUT, это стирает то, что находится в файле.

Четвертая глава на этом заканчивается. Вот задание:

1.Сделать программу в которой нужно сгенерировать случайное число между 1 и 20.
Дать играющему 5 шансов, и после того, как вы ввели число,
программа должна сообщать высоко или низко введённое вами число.

2.Сделать программу, которая бы сохраняла высшие результаты в файл.

Глава 5

Глава ПЯТАЯ. Привет! В этой главе, мы изучим операторы:

WHILE...WEND, Random Access Files, APPEND, GET (file I/O), SUB...END, SUB, PUT(file I/O),
FUNCTION...END FUNCTION, LEN, GOSUB...RETURN, TYPE...END, TYPE,
DIM SHARED, COMMON SHARED, RTRIM$

Чтож, приступим! Мы изучим другой метод выполнения цикла, путь прибавления к концу
последовательного файла, структурного программирования.
Сначала вспомним прошлые способы выполнения цикла.

Как вы помните DO...LOOP и FOR...NEXT, а сейчас мы изучим операторы WHILE...WEND.
Этот оператор очень похож на цикл DO...UNTIL. Вот как выглядит синтаксис этого цикла:

```
 WHILE<условие>
 ...
WEND
```

Скажем, Вы хотите сделать простую игру в предположения.
Всё, что требуется - простой цикл, как в этом примере:

```
 RANDOMIZE TIMER
 PRINT "Игра - угадай!"
 number = INT(RND * 10) + 1'случайное число от 1 до 10
 PRINT "Секретный номер находится в промежутке от 1 до 10"
 WHILE guess <> 10' цикл до предположения = 10
 INPUT "Попыток "; guess
 tries = tries + 1'счётчик попыток
 WEND
 PRINT "Хорошая работа! Вы угадали секретный номер с "; tries; "  попыт(ок),(ки)"
 END
```

Эта программа составлена только в 10 строк!
Вот задание - объясните,
что делает каждая строча(как вы видите на некоторых строчках есть комментарии).
Теперь вы поняли, как действует цикл WHILE...WEND, он простой, и все же полезный!
Мы говорили относительно файлов в Главе 4.
Вы помните команды INPUT и OUTPUT для последовательного входа в файл.
Есть еще один способ добавления к файлу информации, используя команду APPEND.
Команда используется точно так же как и OUTPUT и INPUT:

```
 OPEN "file.txt" FOR APPEND AS #1
```

APPEND действует точно также как и OUTPUT и кроме этого не стирает то,
что уже записано в файле, прежде чем добавлять к этому другой текст.
Это полезно для продолжающихся высоких списков счета и связки других вещей.
Сейчас мы должны изучить структурное программирование.
Очень не удобно в большой программе с тысячами строк и множеством операторов найти проблему!
Поэтому, умные люди в 1960-ых разработали кое-что
называемое "структурным программированием - Ming. "
При использовании этой методики, мы можем использовать "подпрограммы" и "функции"
в создании наших программ, чтобы упорядочить программный текст.

Чтобы создавать и редактировать подпрограмму в QBASIC,
нажимают кнопку на инструментальной панели меню
[E]dit(редактировать) > New [S]UB...(Новая подпрограмма...).
QBASIC спросит у вас относительно имени подпрограммы. Можно давать любое имя.
QBASIC тогда позволит вам редактировать текст подпрограммы.
В тексте подпрограммы используются те же операторы что и в обычной программе,
после того как вы написали подпрограмму, вы можете её посмотреть,
идите к [V]iew на инструментальной панели, выберите [S]UBS.
И затем выберите имя файла вашей подпрограммы в меню.
Если Вы хотите вернуться к главному тексту программу, то,

выберете в этом же меню название вашей программы. Знайте, что переменные,
которые Вы создаете в подпрограмме доступный только в этой же подпрограмме,
а переменные в теле программы доступны только в нём, сделать их общедоступными можно с
использование следующая команда в начале подпрограммы:

```
COMMON SHARED variable$, variable2, variable3!
```

Команда "DECLARE SUB...", объявляет имя подпрограммы и её существование.
(QBASIC размещает эти инструкции в начале вашей программы, как только вы сохраняете её).
Сделать переменную доступной везде можно так (используя в задании переменной оператора SHARED):

```
DIM SHARED array$(100)
```

Удостоверитесь, что любые переменные(называемые "глобальной переменной")
находятся в вашей команде COMMON SHARED, иначе ваша программа будет работать с ошибками.
Передают переменные к подпрограмме подобно этому:

```
SUB DoBox (x1, y1, x2, y2)
```

Чтобы вызвать подпрограмму нужно написать имя подпрограммы и задать численное значение
переменных(если они есть) подобно этому:

```
DoBox 30, 20, 50, 20
```

Если же нет переменных вы просто можете только напечатать название подпрограммы
в основном тексте программы(теле), подобно этому(здесь вызывается сразу 5 попрограмм):

```
Mainloop: CLS
DoGraphics
DoLevel
WaitForKey
```

Как дополнение к подпрограммам есть ещё и функции. Они подобны подпрограммам,
о в основном они используются для вычисления. Создаёте их можно тем же самым путём,
каким вы создали бы подпрограмму, [E]dit > [F]UNCTION(но не SUB).
Редактировать функции можно так же как и подпрограммы. Вот простой пример:

```
FUNCTION Cube(num)
Cube = num * num * num
END FUNCTION

'[главная программа:]

CLS
INPUT "Номер "; number
num3 = Cube(number)
PRINT number; "в кубе = "; num3
END
```

Последний раздел, который мы будем изучать в этой главе - файл прямого доступа.
Они очень полезны для прикладных программ базы данных.
Чтобы сделать программу вы должны определить название вашей TYPE(название) программы и конец

TYPE...END. Имеется короткий пример этого:

```
 TYPE people
 nm AS STRING * 40 ' название программы - 40 символов
 age AS INTEGER ' устанавливает возраст как целое число
 address AS STRING * 60 ' устанавливает адрес в 60 символов
 END TYPE
```

Следующая вещь, которую мы должны изучить прежде,
перед тем как вы откроете файл - использование команды DIM,
для установления типа прямого доступа к файлу. Это делается так:

```
 DIM person AS people
```

Теперь, мы должны научится открывать файл.
Снова используется команда OPEN(как вы помните из предыдущей главы), но теперь мы должны добавиться использования двух команд LEN и DIM. Вот пример:

```
 OPEN "address.dat" FOR RANDOM AS #1 LEN = LEN(person)
```

Это открывает файл прямого доступа "address.dat" с книгой записей для присоединения информации.
Теперь вы должны научится использовать свои переменные для ввода в файл информации.
Пример описан ниже:

```
 INPUT "Дайте название записи "; record
 INPUT "Имя"; person.nm
 INPUT "Возраст"; person.age
 INPUT "Адрес"; person.address
 PUT 1, record, person
```

Как вы видите, вы должны задавать переменной имя массива, а точка, в вашей пременной ТИП.
Тогда, Командой PUT, вы помещаете переменные в файл.
Синтаксис для команды PUT:
PUT [имя файла], [номер записи], [имя переменной]
Это очень просто. Получать массив из файла, используется в основном тот же самый метод, за исключением команды GET .
Команда GET точно имеет тот же самый синтаксис, как команда PUT,
за исключением, того, что из файла читается в массив, который вы определяете.
Есть пример:

```
 INPUT " Представление, которые записи "; record
 GET 1, record, person
 PRINT "Название"; person.nm
 PRINT "Возраст"; person.age
 PRINT "Адрес"; person.address
```

Так используют файл прямого доступа. Назовём, для примера файл RANDOM.BAS.
Файлы прямого доступа являются очень полезным для большого количества прикладных программ, но они очень комплексны.

```
 Personname$ = RTRIM$ (person.nm)
```

Такая команда бы подстроила к концу переменной несколько пробелов.

Задание:

1. Создать простую программу базы данных для адресов и названий(имен)
Использование структурного программирования.

Выскажите своё мнение

Наименование *

e-mail *

Сайт

CAPTCHA Code *

Отправить комментарий

С уважением,

Дилшод Саидов!

Web-site: analytic-73.ucoz.com

E-mail: analytic-73@mail.ru

Mobile phone: (+992) 92 771-38-58