

# **Особенности программирования в Linux СПО на алгоритмическом языке Basic-256**

Преподаватель информатики  
высшей категории **И.М.Балонов**  
и ученик 11 класса МОУ СОШ №22 с  
углубленным изучением иностранных языков  
**Хамзин Марат**

## **Введение**

В 2008 году на территории России началась крупная программа по внедрению свободного программного обеспечения в учебный процесс. В частности в МОУ СОШ №22 г.Перми был поставлен комплект дистрибутивов операционной системы AltLinux. При работе с этой операционной системой сразу бросается в глаза огромное количество средств разработки.

В целом, язык программирования служит двум связанным между собой целям: он дает программисту аппарат для задания действий, которые должны быть выполнены, и формирует концепции, которыми пользуется программист, размышляя о том, что делать.

Первой цели идеально отвечает язык, который “близок к машине”, что всеми основными машинными аспектами можно легко и просто оперировать достаточно очевидным для программиста образом.

Второй цели идеально отвечает язык, который “близок к решаемой задаче”, чтобы концепции ее решения можно было выражать прямо и коротко.

В данную сборку «AltLinux» встроен диалект алгоритмического языка Basic под названием «Basic-256». Данный язык изначально имел название «KidBasic» (детский Basic), то есть при разработке данного языка создатели преследовали цель, обучить детей основам программирования.

Следует заметить, что данный алгоритмический язык является самостоятельной программой и доступен как на операционной системе Linux, так и на операционной системе Microsoft Windows.

Работа написана очень простым и доступным языком и может использоваться и как инструкция, а в целом ряде случаев, и как пошаговый алгоритм работы с «Basic-256». В неё включено довольно много программ, которые могут являться примерами использования этого алгоритмического языка и, во многом облегчат практику применения пакета в педагогической деятельности.

# **Краткое различие операционных систем**

## **Linux СПО и Microsoft Windows**

### **Успехи двух систем**

Две эти операционные системы упорно соперничают за внимание не только пользователей персональных компьютеров, но и на серверах, в государственных учреждениях, офисах, суперкомпьютерах и встроенных системах

На данный момент Windows завоевала пользовательскую аудиторию, около 90% настольных компьютеров работают на Windows. Linux более популярна на web-серверах, вычислительных кластерах и в суперкомпьютерах (50-80 %).

### **Полный доступ или его отсутствие**

Пожалуй, одно из самых существенных различий между Linux и Windows — наличие или отсутствие доступа к исходному программному коду. Linux разрабатывается в соответствии с открытым лицензионным соглашением GNU (GPL), поэтому все пользователи имеют право и возможность просматривать и изменять исходный программный код вплоть до самого ядра, которое служит основой операционной системы Linux. Вы сможете посмотреть исходный код Windows, если только вы не принадлежите к элитной (для многих) группе избранных. Увидеть исходный код ОС Microsoft вам никогда не удастся.

На этот вопрос можно взглянуть с разных точек зрения. Некоторые опасаются, что свободный доступ к исходному коду делает операционную систему и ее программное обеспечение уязвимым для разработчиков вредоносных приложений, которые могут обнаружить в системе лазейки и воспользоваться ими. Другие, наоборот, считают, что свободный доступ к исходному коду ускоряет процесс усовершенствования программного обеспечения как раз для предотвращения атак хакеров.

### **Свободное лицензирование или лицензионные ограничения**

Другое фундаментальное различие между Linux и Windows проходит на уровне лицензий на использование программного обеспечения. Открытое лицензионное соглашение на ОС Linux позволяет свободно модифицировать программное обеспечение, использовать, публиковать его от своего имени и даже продавать — главное, чтобы исходный код по-прежнему оставался открытым. К тому же, GPL позволяет загрузить одну копию дистрибутива или приложения Linux и установить ее на неограниченном количестве компьютеров. Лицензия Microsoft ничего подобного не допускает. Пользователь ограничен количеством приобретенных лицензий, и если у вас имеется десять лицензий, вы можете законно установить приобретенную операционную систему или приложение только на десять компьютеров.

## Пользовательская поддержка в Интернете или платная поддержка технических специалистов

Именно этот аспект отвращает от Linux корпоративных пользователей — и совершенно напрасно. Техническую поддержку по Linux можно получить в огромном сообществе пользователей — на форумах, в поисковиках и на сотнях специализированных веб-сайтов. А при большом желании можно купить сертификат на техническую поддержку у одного из крупных поставщиков Linux типа Red Hat и Novell.

Правда, пользовательская техническая поддержка в Интернете тесно сопряжена с проблемой времени. Иногда в ответ на сообщение с рассказом о возникшей проблеме можно получить сотни откликов за какие-нибудь десять минут, но порой этих откликов приходится дожидаться часами, днями и даже неделями. Очень часто все зависит от случая. И все же, большинство проблем, связанных с Linux, уже изучено и задокументировано, так что решение можно найти относительно быстро.

С другой стороны, посмотрите, как организована техническая поддержка Windows. Конечно, здесь можно пойти тем же путем и поискать ответы на свои вопросы на сайтах, форумах, в рассылках и так далее — подобного материала по Windows в Интернете ничуть не меньше, чем по Linux. А можно купить сертификат на техническую поддержку непосредственно у Microsoft. Руководители большинства корпораций легко попадаются в ловушку мнимого ощущения безопасности, которую обеспечивает наличие такого сертификата. Но зависеть от этого сертификата совсем не обязательно.

## Полная или частичная аппаратная совместимость

Одна проблема Linux, постепенно отходящая в прошлое, — это проблема аппаратной совместимости. Несколько лет назад для успешной установки Linux на настольном компьютере все компоненты системы приходилось подбирать вручную, иначе не было никакой гарантии, что ОС заработает. Сегодня на любой компьютер или ноутбук можно установить хотя бы один, а чаще несколько дистрибутивов Linux, которые будут работать на все сто процентов. Но есть, конечно, и исключения. Например, режим ожидания/сна до сих пор работает некорректно на многих ноутбуках, несмотря на то, что разработчики бьются над этой проблемой уже давно.

И в то же время, с Windows совместимо практически любое оборудование.

## Командная строка или ее отсутствие

Как бы далеко операционные системы Linux ни зашли в своем развитии, и как бы ни был великолепен их графический пользовательский интерфейс, командная строка всегда будет оставаться незаменимым инструментом для выполнения любых административных задач. Правда, для конечного пользователя это не так актуально. Человек может годами пользоваться операционной системой Linux и даже не прикоснуться ни разу к командной строке. То же самое и с Windows. Здесь командной строкой пользоваться можно, но далеко не так широко, как в Linux. К тому же, Microsoft изо всех сил старается спрятать командную строку от пользователей: добраться до нее можно, только если запустить средство «Выполнить» (Run) и ввести cmd.

# Безопасность

В Linux (как и во всех других UNIX-подобных системах) всегда присутствовало чёткое разделение пользовательских прав. Имеется только одна учётная запись системного администратора («суперпользователя») — *root*. Этот пользователь может выполнять ничем не ограниченные действия над системой: изменять настройки, устанавливать и удалять программы, изменять системные файлы, останавливать работу системы или отдельных ее компонентов. Поэтому постоянно работать под *root* категорически не рекомендуется: можно испортить конфигурацию или системные файлы — последствия будут необратимыми. И имеются учётные записи обычных пользователей: они могут только изменять личные настройки (внешний вид, настройки программ), и выполнять операции с файлами только в пределах своего домашнего каталога (или в других каталогах, если разрешит *root*). Устанавливать программы можно только в свой домашний каталог или в те каталоги, где у пользователя есть разрешение на запись данных. Отдельный разговор касается вирусов в обеих ОС. На данный момент во всемирной паутине насчитывается более миллиона вирусов для Windows (это не считая всех вирусов в каждом семействе). За 17 лет существования Linux на него написано примерно 23 более или менее серьезного вируса.

# История алгоритмического языка Basic

## Рождение

В 1963 году преподаватели Дартмутского Колледжа Джон Кемени и Томас Куртиц придумали алгоритмический язык Бейсик. Под их руководством группа студентов реализовала этот проект. С течением времени начали появляться многочисленные диалекты этого языка, «изначальный» диалект называли Dartmouth BASIC.

Особенностью проектировки Бейсик стала возможность написания программ с использованием терминалом с разделением времени. Языки того времени были достаточно сложны, Бейсик создавался для решения проблем, связанных со сложностью других языков. Бейсик уже изначально предполагался, как язык для более «простых» пользователей, которые не столько были заинтересованы в скорости программ, сколько в возможности использовать компьютер для решения собственных задач.

Проектировщики языка руководствовались следующими принципами. Язык должен:

1. быть простым в использовании для начинающих;
2. быть языком программирования нового назначения;
3. предоставлять возможность расширения функциональности, доступную опытным программистам;
4. быть интерактивным;
5. предоставлять ясные сообщения об ошибках;
6. быстро работать на небольших программах;
7. не требовать понимания работы аппаратного обеспечения;
8. защищать пользователя от операционной системы.

Основанием языка послужили частично Фортран II и частично Алгол-60, были произведены некоторые добавления, делающие язык удобным для работы в режиме разделения времени и, позднее, обработки текста и матричной арифметики. Сначала Бейсик был реализован на майнфрейме GE-265 с поддержкой множества терминалов. Вопреки распространенному мнению, Бейсик изначально был компилируемым языком.

## Взрывной рост

Язык уже использовался на нескольких миникомпьютерах, но его настоящеее распространение началось с его появления на микрокомпьютере Altair 8800. Многие языки программирования были слишком большими по меркам тех времен, поэтому рядовые пользователи не могли их использовать. Для машин с таким медленным носителем, как бумажная лента (позднее – аудиокассета) и без подходящего текстового редактора Бейсик стал настоящим решением проблемы.

В 1975 году Майкрософт (в то время это были лишь двое – Билл Гейтс и Пол Ален) при участии Монте Давидова выпустила Altair BASIC. Потом его версии появились на другой платформе под лицензией. Через небольшой промежуток времени появились

миллионы копий и вариантов языка, один из них, Applesoft BASIC, стал стандартным языком на Apple II. Для операционной системы CP/M был создан собственный диалект BASIC-80, который надолго вперед определил развитие языка.

В 1979 году Майкрософт обсуждала с поставщиками компьютеров лицензию интерпретатора Бейсик на их компьютерах. ROM BASIC был включен в ПЗУ IBM PC – этот компьютер мог автоматически загружаться в Бейсик. В то время IBM не придавала большого значения персональным компьютерам (основным полем ее деятельности были мейнфреймы), поэтому она разрешила Майкрософт продавать интерпретатор отдельно. Это сделало Майкрософт пионером в среде ПО нового поколения – не привязанного к конкретной аппаратуре и поставляемого отдельно от компьютера.

## Зрелость

В этот период было создано несколько новых версий Бейсика. Майкрософт продавала несколько версий Бейсик для MS-DOS/PC-DOS, такие как BASIC-A, GWBASIC и Quick BASIC. В 1985 году Borland, которая прославилась, благодаря своему Turbo Pascal, выпустила Turbo BASIC 1.0 (позже его наследники продавались компанией PowerBASIC). У Бейсика появилось множество версий для домашних компьютеров, которые в свою очередь включали расширения для работы с графикой, звуком, команды ОС, а также средства структурного программирования. Некоторые другие языки использовали в качестве основы всем известный синтаксис Бейсика, но в свою очередь строили совершенно иную систему, например GRASS.

Начиная с конца 80-х, Бейсик становился все менее удобным для программирования, так как начали появляться новые, более сложные компьютеры с новыми возможностями, такими как графический интерфейс. Бейсик начал терять былую популярность, хотя все еще множество его версий использовалось и продавалось.

Visual BASIC от Microsoft вдохнул новую жизнь в этот язык программирования. Не смотря на множество привычных ключевых слов, Visual BASIC был совсем не похож на привычный всем BASIC. Постепенно он стал одним из самых часто используемых языков на Microsoft Windows. Microsoft использовал Word Basic до появления Word 97. Visual Basic for Applications (VBA) в 1993 году был встроен в Excel 5.0, в 1995 – в Access 95, в 1997 – во все инструменты, входящие в пакет Microsoft Office. В свою очередь Internet Explorer 3.0 и выше, Microsoft Outlook включали интерпретатор языка VBScript. OpenOffice.org также включает в себя интерпретатор Бейсика.

# Описание языка

Многие элементы синтаксиса напоминают Fortran. Так как язык задумывался для обучения, то многие элементы его максимально просты. Ключевые слова, как и в других языках программирования, взяты из английского языка. В языке существуют два основных типа данных: строки и числа. С появлением Visual Basic и различных его модификаций, в языке появились другие типы данных, и дополнения, типичные для современных языков программирования. Объявление переменной – первое ее использование, не требует специальной секции.

Ранние версии Бейсика сильно отличаются от современных, на данный момент, они практически не используются.

## Ранние версии

В ранних версиях обязательной была нумерация строк. Стандартной стала нумерация с шагом 10. Тогда Бейсик не имел полноценного редактора кода, поэтому для вставки новой строки нужно было дописать строку с номером, находящимся в диапазоне между номерами двух строк. Например, чтобы дописать строку между строками 50 и 60, нужно было написать строку с номером 55. Типичная строка из программы выглядела так:

**10 PRINT «Hello, world!»**

Номера строк имели функцию меток для оператора GOTO. Для этого оператора наличие нумерации строк делало возможным безусловный переход к выбранной строке. Использование нумерации и оператора GOTO было необходимым при программировании на Бейсике, это, в свою очередь, способствовало плохой структуре кода и зачастую запутывало самих программистов.

В качестве конца оператора применяется перевод строки. При необходимости разместить а одной строке несколько операторов, между ними ставится двоеточие. Имена переменных в большинствеialectов было принято ограничивать тремя символами. Это еще более сильно усложняло понимание кода. Чтобы ввести переменную, нужно было после имени указать знак доллара (\$). Переменная без этого знака являлась числовой. Существовало еще несколько модификаторов, относящихся к числовому типу: знак процента (%) – целый тип, знак восклицания (!) – обычная точность, октоторп (#) – двойная точность (для дробей). Не во всех версиях языка были введены числовые модификаторы.

Выражения на языке Бейсик сходны с большинством других языков программирования. Присутствуют все базовые управляющие структуры: условный оператор (IF..THEN..ELSE) и циклы (FOR..WHILE). Функции, определяемые пользователем (участки кода для многократного использования), можно было использовать двумя способами: определяемые функции и подпрограммы.

Подпрограммы не являлись аналогом функций таких языков, как Си или Паскаль, так как не имели возможности принимать и/или возвращать параметры. Это было возможно только с помощью переменных. Подпрограммы обычно писали с заведомо большей нумерацией (чем предполагается) в конце модуля. Например, нумерация подпрограммы могла начинаться с 5000, даже если в программе пара сотен строк. Подпрограммы ничем не отличались от какого-либо другого участка кода, так как не имели ни заголовков, ни имен. Вызов подпрограммы осуществлялся с помощью оператора GOSUB (метка). Этот оператор практически идентичен оператору GOTO. За исключением того, что возврат в точку вызова происходил автоматически при достижении в подпрограмме ключевого слова RETURN. GOSUB, как и GOTO, - способствовали лишь непонятной структуре кода. Отныне программисты писали весь код в виде подпрограмм, начало программы выглядело так:

```
30 GOSUB 170: GOSUB 300
40 GOSUB 730: GOSUB 1410: GOSUB 840
50 GOSUB 950
60 IF F$= " " THEN GOSUB 860: GOSUB 380: GOSUB 840: GOTO
50
70 ON VAL(F$) GOTO 80,110,130
80 GOSUB 990: IF T=1 THEN 50
90 GOSUB 860: GOSUB 910: GOSUB 840
100 GOTO 50
110 GOSUB 1000: IF T=1 THEN GOSUB 1080: GOSUB 1120:
NG=0: GOTO 40
120 GOSUB 860: GOSUB 900: GOSUB 840: GOTO 50
130 GOSUB 980: IF T=1 THEN 50
140 GOSUB 860: GOSUB 920: GOSUB 840: GOTO 50
```

Так как возможности давать подпрограммам имен не имелось (и вызывать по ним), невозможность передачи параметров, использование только числовых меток – в больших программах могло вызвать только путаницу.

В некоторых диалектах Бейсика имелась возможность подгружать подпрограммы по мере необходимости во время выполнения основного кода оператором CHAIN. Это была одна из самых полезных возможностей Бейсика, так как она в разы упрощала основной код, позволяла разбить большой проект на модули и вызывать их по мере необходимости.

Редактора кода в ранних версиях Бейсика не было. Был лишь интерпретатор. При его запуске, запускался диалоговый режим ввода команд. Для работы в этом режиме предусматривались специальные команды, которые не включались в основной код. Это команды по управлению средой кода, такие как: LIST -<диапазон строк>- – вывод листинга программы на экран (LLIST – на принтер), Save –<имя файла>- - сохранение текста программы в файл, LOAD -<имя файла>- - загрузка кода в память и т.д. Запуск программы производился с помощью команды RUN. В большинстве интерпретаторов Бейсика была строка в нижней части экрана с этими (и другими) командами.

Команда, которая начиналась с номера строки, воспринималась, как часть программы. Если строка оператора вводилась без номера, она запускалась сразу после нажатия клавиши ENTER. Обучающимся на Бейсике для начала предлагалось поэкспериментировать с командами, результат работы которых был виден сразу. Например, команда PRINT.

Как таковое, редактирование строки было невозможно. Чтобы отредактировать строку, надо было полностью переписать ее. Удаление строк производилось программой DELETE <диапазон строк>. Команда RENUM позволяла восстановить шаг 10 для всех строк. Часто возникала ситуация, что шага 10 не хватало, так как между двумя строками можно было вписать лишь 9 строк, тогда команда RENUM стала необходимой. Все строки переименовывались с шагом 10, так же корректировались все строки с командой GOTO для соответствующих строк.

Фрагмент типичной программы на Бейсике:

```
10 CLS 'Очистка экрана
20 PRINT «добро пожаловать!» 'Заголовок в первой строке
30 'Цикл, выводящий линию под заголовком, на всю ширину
экрана
40 FOR I=1 TO 80
50 PRINT «=»;
60 NEXT I
65 'Ввод символьных данных от пользователя (комментарий
добавлен после ввода нижних строк)
70 INPUT «Имя: »,N$
80 INPUT «Фамилия: »,S$
90 INPUT «Отчество: »,T$
95 'Вырезаем копию первых символов из имени и отчества
100 N2$=LEFT$(N$,1,1)
110 T2$=LEFT$(T$,1,1)
120 'Выводим результат
130 PRINT «Ваше имя кратко: «;S$;» «;N2$;». «;T2$»..»
140 INPUT «Повторить программу? (Y/N) »,$U
150 IF U$="Y" THEN GOTO 10
160 END
```

Точка с запятой выполняет функцию показателя перехода программы на новую строку после оператора PRINT. Ключевое слово END? Указывающее завершение программы, - необязательно, но оно может понадобиться, если имеется секция подпрограмм.

## Поздние версии

Появление Visual Basic, созданного для разработки программ с графическим интерфейсом пользователя для Windows, во многом улучшило ситуацию. Бейсик был существенно расширен, как язык программирования, отпала необходимость частого использования команды GOTO. В качестве меток для команды GONO теперь может использоваться любая константа, а не только целые числа. Подпрограммы с GOSUB теперь не используются, вызов процедур теперь осуществляется по имени функции. Тип переменной теперь указывать не обязательно, но его указание не считается ошибкой. Практически все недостатки, свойственные ранним версиям, были устранены на современных диалектах. Программирование на Бейсик стало практически так же удобно и гибко, как и при использовании аналогов.

Существует множество различных современных компиляторов для Basic. В состав платформы Microsoft.Net входит компилятор VisualBasic.Net. Помимо этого, Бейсик хорошо применяется как основа для различных скриптовых языков. Например, написание макросов в пакете Microsoft Office выполняется на специализированном диалекте VBA.

# Basic-256 и особенности его синтаксиса и семантики

«Basic 256»-программы состоят из утверждений, разделенных строками, которые исполняются в своем порядке.

## Числовые константы

Числовыми константами являются любые численные символы. Чтобы обозначить отрицательно число, перед числовым знаком надо поставить минус.

## Строка констант

Константы от нуля и более символов заключаются в кавычки ("").

## Переменные

Переменные, которые приводят числовые значения, должны начинаться с буквы и могут включать любое количество численно-буквенных символов. Они могут быть использованы, как заменители числовых констант.

Для переменных, которые приводят комбинацию значений используются те же правила, но комбинация должна заканчиваться знаком доллара (\$). Они могут быть использованы, как заменители строки констант.

## Множества

Множества, выделяемые с помощью команды DIM могут иметь цифровой вид или вид строки данных. Доступ к конкретным элементам множества осуществляется с помощью квадратных скобок, заключающими целые числа, начиная с нуля. Например:

```
myarray [4]
```

будет доступен пятый элемент в «myarray».

## Безымянные множества

Безымянные множества представляют собой набор числовых значений, разделенных запятыми и заключенными в круглые скобки.

Безымянное множество может быть использовано вместо множества переменной или оно может быть использовано для присвоения множеству переменной, а именно:

```
dim myarray (4)
```

```
myarray = (1, 2, 3, 4)
```

# Знаки

Знаки +, -, \*, / используются для выполнения сложения, вычитания, умножения и деления соответственно. Допустимые действия выполняются для числовых констант и\или переменных.

= Знак используется для задания переменных, а также для проверки равенства.

+ Знак может быть использован для выполнения объединения в любом сочетании строковых констант и переменных в строку.

: Знак может разделять несколько операторов в одной строке.

# Команды

## Abs

abs (expression)

Возвращает абсолютное значение выражения.

Например строка a=abs(-256) выдаст значение 256

## Ceil

ceil (expression)

Возвращает наименьшее целое значение выражения, которое больше или равно этому выражению.

## Circle

circle x,y,r

Круг с центром с координатами (x;y) и радиусом r в текущем цвете.

## Clg

Очищает графический экран окна вывода.

## Close

Закрывает открытый файл/программу. Если файл/программа не является открытой, команда ничего не делает.

## Cls

Очищает текст из окна вывода.

## Color

color [colorname]

Устанавливает цвет.

## Cos

cos (expression)

Вычисляет косинус выражения (в радианах).

## **Dim**

dim (integer)

Позволяет задать переменную. Создает множество с данной переменной.

## **End**

Завершение работы программы.

## **Fastgraphics**

Fastgraphics режим означает, что графический дисплей не обновляется до тех пор, пока не будет дана команда REFRESH. Используется для значительного ускорения сложной анимации.

## **Floor**

floor (expression)

Возвращает наибольшее целое значение выражения, которое меньше или равно этому выражению.

## **For/Next**

for variable = expression1 to expression2 [step expression3]

Команды FOR и NEXT используются в сочетании, чтобы использовать команду или группу команд определенное количество раз. После каждой следующей команды переменная увеличивается на значение шага (по умолчанию =1).

## **Goto**

goto label

Переход на указанную метку.

## **Gosub**

gosub label

Переход к указанной метке. Возврат в точку вызова происходит автоматически при достижении ключевого слова RETURN.

## **Instr**

instr string1, string2

проверяет содержится ли string2 в string1. Если содержитя, то программа показывает на каком месте string2 содержится в string1. В противном случае, программа показывает 0.

## **Int**

int (expression)

int (string expression)

Преобразование в целое. Программа преобразовывает числа с плавающей точкой или конвертирует строку в целое.

## **If/Then**

if booleanexpr then statement

Оценивает booleanexpr. Если верно, то statement выполняется. Если неверно, то продолжается выполнение этой строки или начинается выполнение следующей.

## **Input**

input string, stringvar

Окно ввода текста пользователем.

## **Key**

Ставит число сразу после нажатия клавиши.

Например:

```
a = key  
if a = 47 then print a
```

## **Length**

length string

Показывает число символов в строке.

## **Line**

line x0, y0, x1, y1

Чертит линию от точки с координатами (x0;y0) до точки с координатами (x1;y1).

## **Mid**

mid string, start character, length

Показывает все символы с указанного по номеру символа по заданной длине.

## **Open**

open filename

Открывает указанный файл. Может быть указано как имя файла, так и путь до файла. Нельзя открыть сразу два файла, при открытии второго файла, первый будет закрыт.

## **Pause**

pause seconds

Останавливает выполнение всех команд в течение указанного времени.

## **Plot**

plot x, y

Изменение пикселей, расположенных в координатах (x;y), в график в текущем цвете.

## **Rem**

rem

Команда позволяет добавить комментарий в код программы, который не будет выполняться при выполнении программы.

## **Reset**

reset

Очищает открытый в данный момент файл. Все данные будут потеряны.

## **Sin**

sin (expression)

Вычисляет синус данного выражения в радианах.

**Tan**

tan (expression)

Вычисляет тангенс выражения в радианах.

**String**

string

Возвращает значение строки.

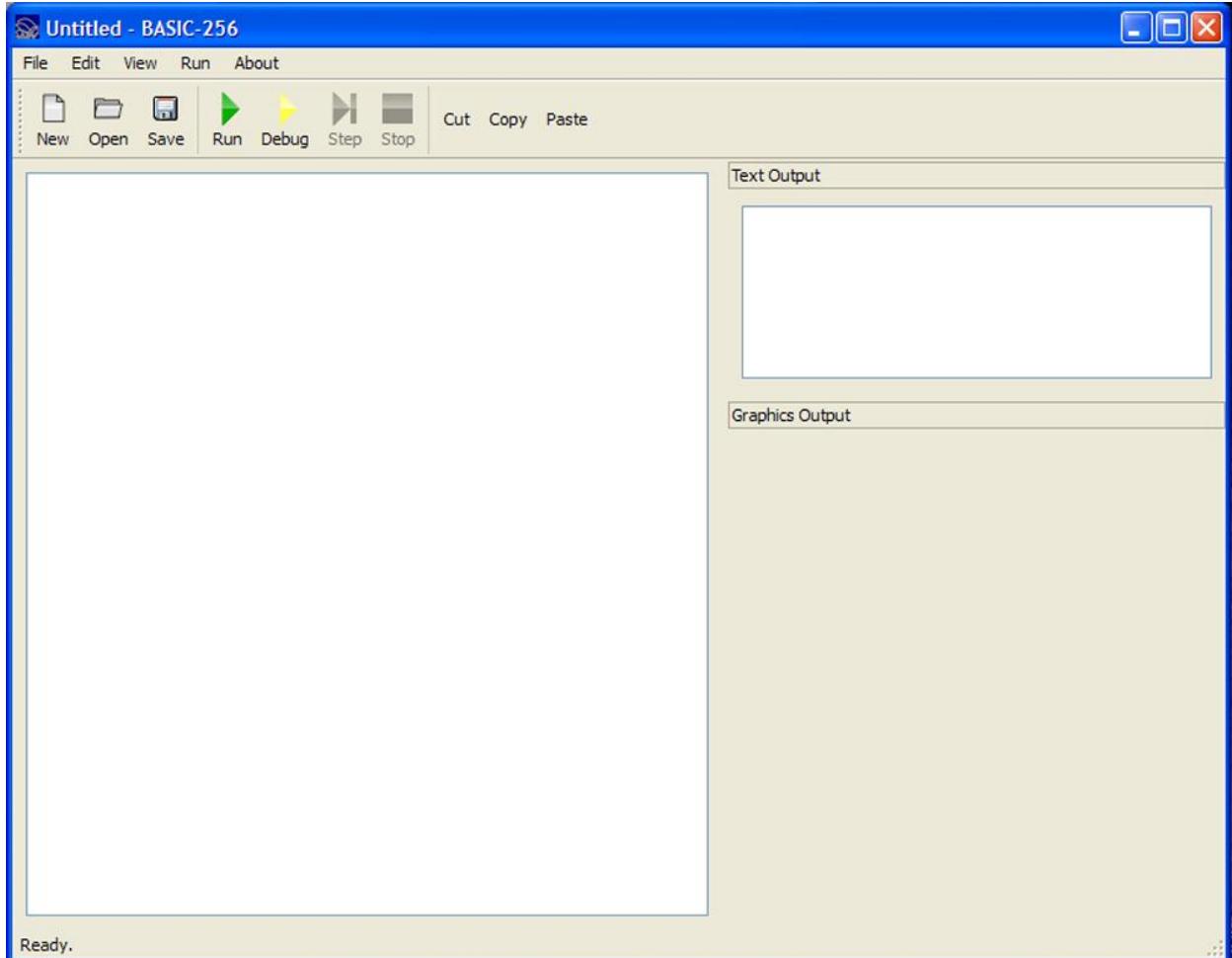
**Write**

write string

Пишет строку до конца открытого в данный момент файла.

# Практическая работа в Basic-256

Главное окно программы:



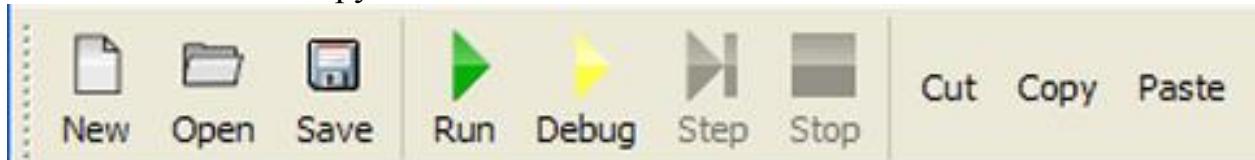
Главное меню:

File Edit View Run About

Подменю:

- File (Файл)
- Edit (Правка)
- View (Вид)
- Run (Запуск)
- About (О программе)

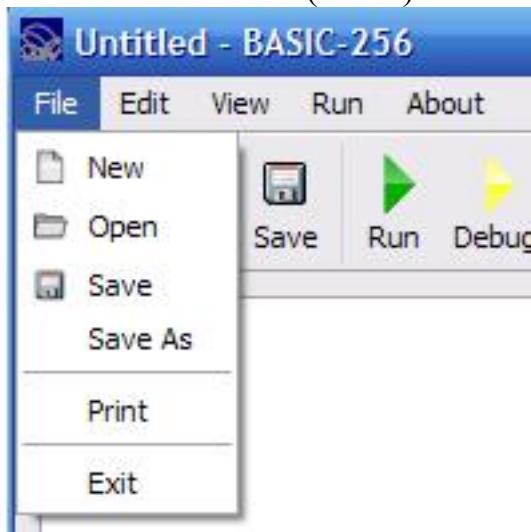
Главная панель инструментов:



Кнопки:

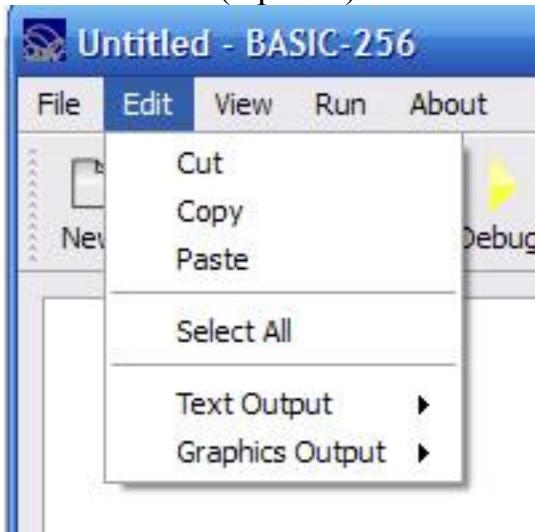
- New (Новый код)
- Open (Открыть ранее сохраненный программный код)
- Save (Сохранить программный код)
- Run (Запуск)
- Debug (Отладка)
- Step (Переместиться на один шаг вперед при выполнении программы)
- Stop (Остановить)
- Cut (Вырезать выделенный элемент кода)
- Copy (Скопировать выделенный элемент кода)
- Paste (Вставить элемент кода)

Меню «File» (Файл):

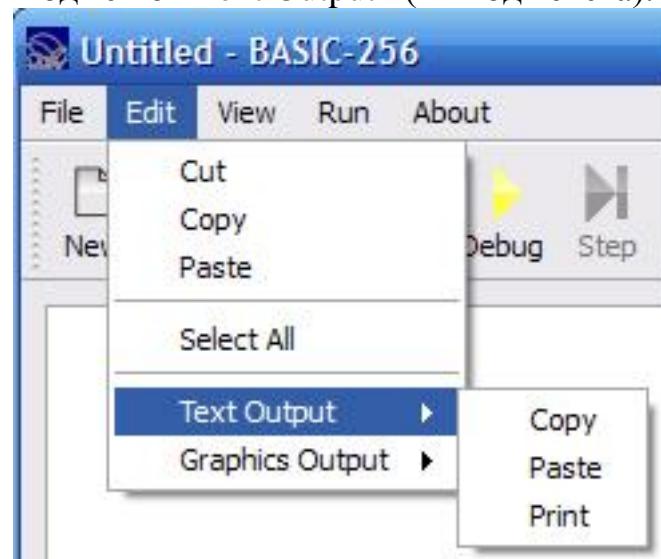


- New (Новый код)
- Open (Открыть ранее сохраненный программный код)
- Save (Сохранить программный код)
- Save as (Сохранить программный код под определенным именем)
- Print (Распечатать)
- Exit (Выйти из программы Basic-256)

Меню «Edit» (Правка):

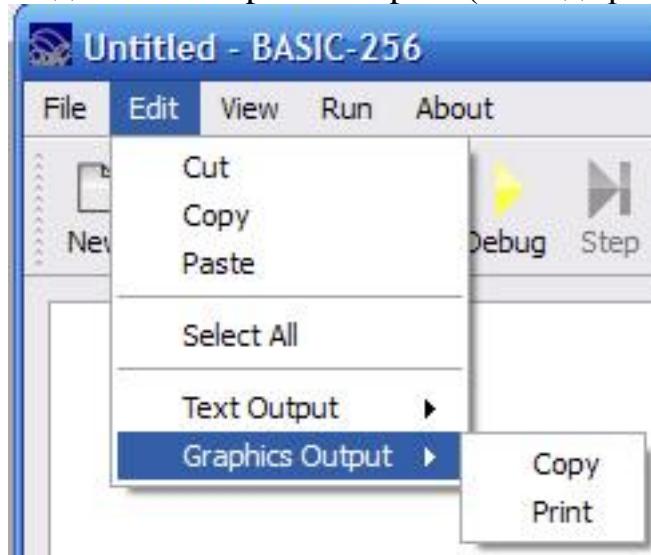


- Cut (Вырезать часть или весь программный код)
- Copy (Копировать часть или весь программный код)
- Paste (Вставить ранее вырезанную или скопированную часть кода)
- Select All (Выделить все)
- Подменю «Text Output» (Вывод текста):



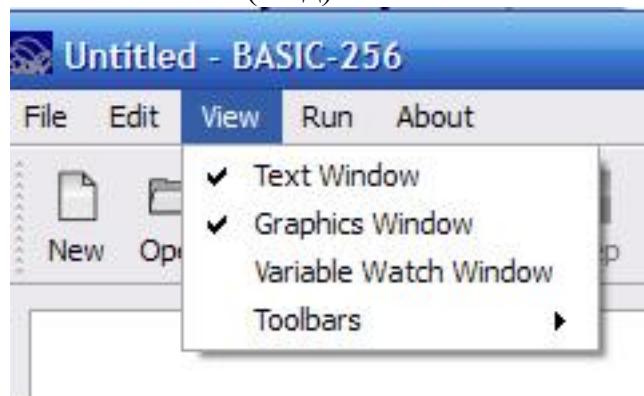
- - Copy (Копировать часть или весь программный код)
  - Paste (Вставить ранее вырезанную или скопированную часть кода)
  - Print (Распечатать)

- Подменю «Graphics Output» (Вывод графики):



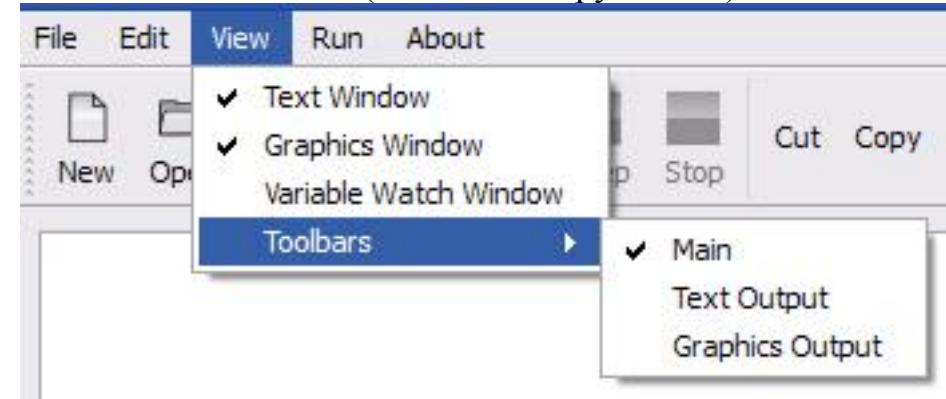
- - Copy (Копировать часть или весь программный код)
  - Print (Распечатать)

#### Меню «View» (Вид):



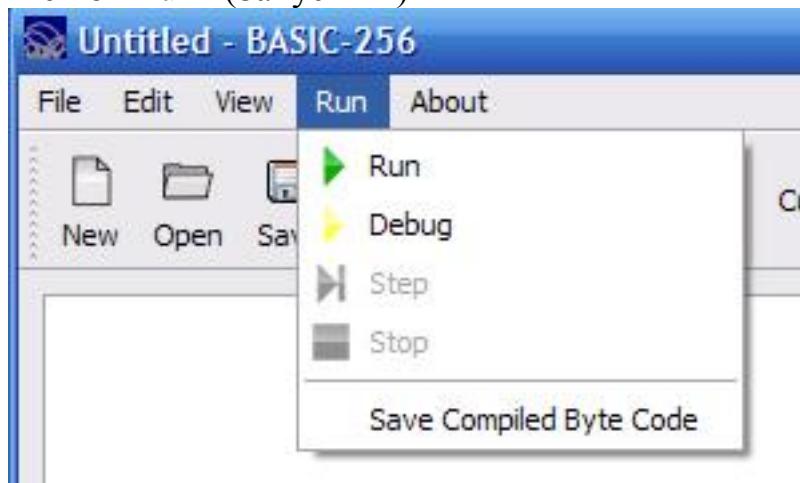
- Text Window (Окно текста)
- Graphics Window (Окно графики)
- Variable Watch Window (Окно введенных переменных)

- Подменю «Toolbars» (Панели инструментов):



- Main (Главное окно)
- Text Output (Вывод текста)
- Graphics Output (Вывод графики)

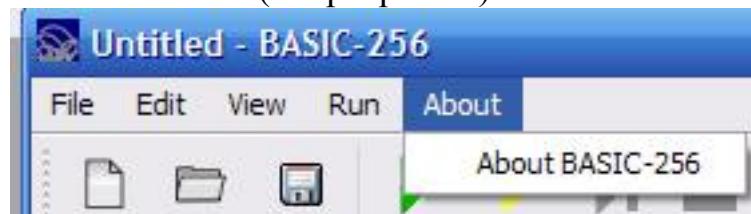
Меню «Run» (Запустить)



Run (Запустить программу)

- Debug (Отладка)
- Step (Переместиться на один шаг вперед при выполнении программы)
- Stop (Остановить выполнение программы)
- Save Compiled Byte Code (Сохранить скомпилированный программный код)

Меню «About» (О программе)



- About BASIC-256 (О программе BASIC-256)

# Примеры программ, выполненных на алгоритмическом языке «Basic-256»

## Программа «Цилинды»

Данная программа предназначена для вычисления площади и объема определенного цилиндра.

Программный код выглядит следующим образом:

```
cls  
*d =  
*h =  
s = 3.14 * d ^ 2 / 2 + 3.14 * d * h  
v = 3.14 * d ^ 2 * h / 4  
print s  
print v  
end
```

\*Недостатком данного алгоритмического языка является невозможность введения командой **Input** численного значения, то есть значения диаметра и высоты цилиндра приходится вводить непосредственно в программный код.

Возьмем произвольные значения переменных:

```
cls  
d = 5  
h = 15  
s = 3.14 * d ^ 2 / 2 + 3.14 * d * h  
v = 3.14 * d ^ 2 * h / 4  
print s  
print v  
end
```

Вычисления показывают, что:

```
s = 150.72  
v = 125.6
```

В данном случае  $\pi=3.14$ . Для получения результатов более высокой точности берется более точное значение  $\pi$ .

## Программа «Графики тригонометрических функций»

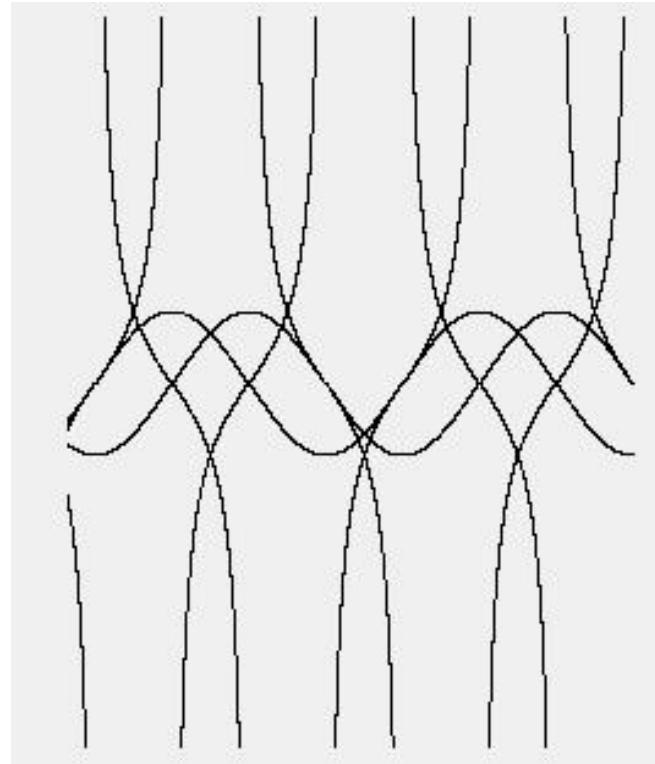
Данная программа предназначена для построения графиков тригонометрических функций, а именно синуса, косинуса, тангенса.

Программный код выглядит следующим образом:

```
clg  
  
fastgraphics  
for x = .5 to 5 * 3.1415962 step .001  
y = sin(x)  
plot int(20*x), 150 + int(30*y)  
next x  
  
refresh  
  
for x = .5 to 5 * 3.1415962 step .001  
y = cos(x)  
plot int(20*x), 150 + int(30*y)  
next x  
  
refresh  
  
for x = .5 to 5 * 3.1415962 step .001  
y = tan(x)  
plot int(20*x), 150 + int(30*y)  
next x  
  
for x = .5 to 5 * 3.1415962 step .001  
y = 1/tan(x)  
plot int(20*x), 150 + int(30*y)  
next x  
  
end
```

При желании в самих функциях можно изменить амплитуду и сдвиг, изменив программный код.

При данных функциях программа выводит следующее графическое изображение



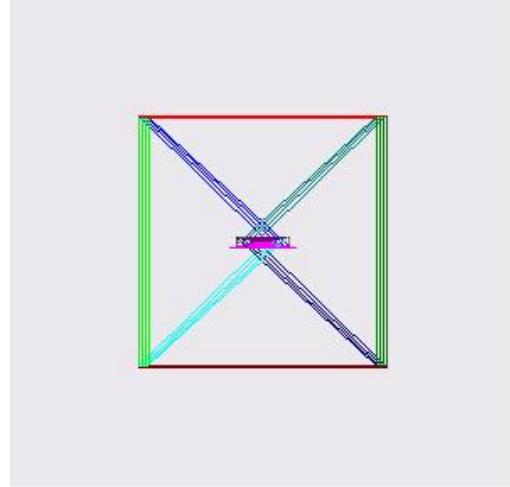
## Графическая программа «Прямоугольники»

Данная программа позволяет в полной мере оценить графические возможности данного языка. Она прорисовывает движущиеся прямоугольники в заданном цикле. В данном случае за основу взят цикл, равный ста.

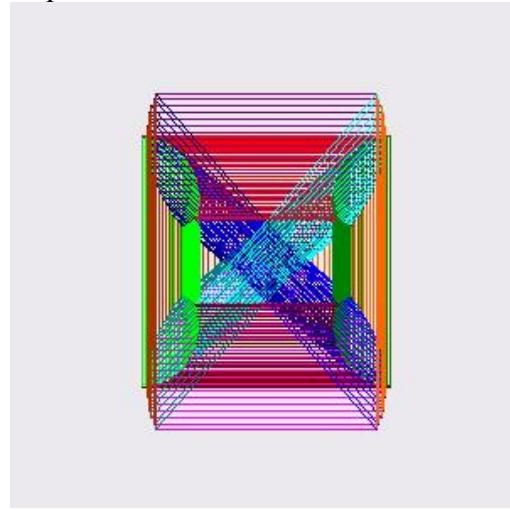
Программный код:

```
clc  
x1=150  
y1=150  
q=0.055  
m=0.0  
for j = 1 to 100  
m=m+q  
x1=x1+4.0*cos(m)  
y1=y1+4.0*sin(m)  
x2=300-x1  
y2=300-y1  
x3=x1/2.0  
y3=y1/2.0  
x4=300-x3  
y4=300-y3  
color purple  
line x1,y1,x2,y1  
color orange  
line x1,y1,x1,y2  
color darkpurple  
line x1,y2,x2,y2  
color darkorange  
line x2,y1,x2,y2  
color blue  
line x3,y3,x1,y1  
color darkblue  
line x2,y2,x4,y4  
color cyan  
line x3,y4,x1,y2  
color darkcyan  
line x2,y1,x4,y3  
color green  
line x3,y3,x3,y4  
color red  
line x3,y3,x4,y3  
color darkgreen  
line x4,y3,x4,y4  
color darkred  
line x3,y4,x4,y4  
next j
```

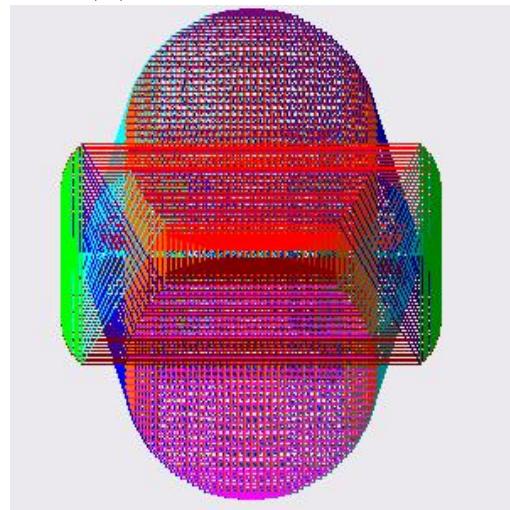
Начало цикла



Середина цикла



Конец цикла



## Программа «Элементы прогрессии»

Данная программа вычисляет все элементы прогрессии с заданным числом элементов, заданным первым членом прогрессии и знаменателем прогрессии.

Программный код:

```
print "Elements of progression"
dim a(11)
a={1,2,3,4,5,6,7,8,9,10}
a[1] = 2
print a[1]
q=3
i=2
M:
a[i]=a[i-1]*q
print a[i]
i = i+1
if i <= 10 then goto M
end
```

Результат вычислений:

Elements of progression	
2	
6	
18	
54	
162	
486	
1458	
4374	
13122	
39366	

## Программа «Уравнение AX=B»

Данная программа вычисляет “x” из уравнения “AX=B”, с заданными параметрами “A” и “B”.

Программный код:

```
print "Equation AX=B"  
a=20  
b=10  
if a = 0 then goto L  
x=b/a  
print x  
goto M  
J:  
if b = 0 then goto J  
print "Net resheniy"  
goto M  
L:  
print "Reshenit beskonechnoe mnojestvo"  
M:  
end
```

Результаты вычислений:

```
Equation AX=B  
0.5
```

## Программа «Цикл – Пока»

Группа операторов, называемая "телом цикла" будет выполняться пока истинно условие цикла. Выход из цикла произойдет, когда условие перестанет выполняться.

	0.782609
Программный код:	0.791667
	0.8
cls	0.807692
print "Cycle while"	0.814815
K=1	0.821429
J:	0.827586
if K>50 then end	0.833333
B=K/(K+5)	0.83871
print B	0.84375
K=K+1	0.848485
if K<=50 then goto J	0.852941
	0.857143
Результаты вычислений:	0.861111
	0.864865
Cycle while	0.868421
0.166667	0.871795
0.285714	0.875
0.375	0.878049
0.444444	0.880952
0.5	0.883721
0.545455	0.886364
0.583333	0.888889
0.615385	0.891304
0.642857	0.893617
0.666667	0.895833
0.6875	0.897959
0.705882	0.9
0.722222	0.901961
0.736842	0.903846
0.75	0.90566
0.761905	0.907407
0.772727	0.909091

## Обсуждение результатов

Достаточно сильной стороной транслятора является графическая часть, в которой расширен набор цветов, засчет введения «dark» оттенков, которые в то же время не совпадают со стандартными для языка Basic.

Отсутствие большинства современных операторов и функций делает невозможным использование существующих библиотек программ, написанных на других диалектах языка. Адаптация требует от учителя и учеников дополнительных усилий.

## Вывод

Таким образом нами был рассмотрен транслятор Basic-256, входящий в комплект ПСПО, оценены его возможности, сильные и слабые стороны. Можно считать, что указанный транслятор, обладая достаточно современным интерфейсом, может оказаться пригодным для использования его в учебном процессе на начальных этапах изучения программирования.

## Список источников

Печатные издания:

1. В.Ф.Шаньгин, А.Я.Пьянзин «Диалоговый язык «Бейсик»;
2. В.М.Заварыкин, В.Г.Житомирский, М.П.Лапчик «Техника вычислений и алгоритмизация»;
3. Ю.Л.Кетков «Диалог на языке бейсик для мини- и микро-ЭВМ».

Интернет ресурсы:

1. <http://kidbasic.sourceforge.net/> - официальный сайт языка «Basic-256»;
2. <http://ru.wikipedia.org/> - «Википедия» свободная энциклопедия.