Altair BASIC

Реализация языка программирования [BASIC](http://progopedia.ru/language/basic/)

Altair BASIC — это интерпретатор языка программирования BASIC, который исполнялся на MITS Altair 8800 и соответствующих компьютерах с шиной S-100. Представлял собой первый программный продукт компании Microsoft (старое название Micro-Soft).

Altair BASIC представлял собой новую веху в истории программ – первый язык программирования для первого персонального компьютера и первый программный продукт компании Microsoft на пути к мировому лидерству в разработке программного обеспечения.

Интерпретатор Altair BASIC разрабатывался основателями Microsoft Биллом Гейтсом и Полом Алленом при участии Монте Давидова. Размер интерпретатора всего 4 Кб, который оставлял достаточный участок памяти для программ, написанных на BASIC.

Однако Altair BASIC был также источником противоречия в конце 1970-ых. Тогда большинство владельцев компьютеров обменивались программами без мысли о лицензировании и покупки. Когда Altair BASIC был завершен, его захотели продать. Но Джону Драперу удалось заполучить перфоленту с Altair BASIC и начал свободно распространять копии. Что привело к разгневанному открытому письму Билла Гейтса к компьютерному сообществу, в котором осуждалось пиратство.

Основные версии Altair BASIC — оригинальный 4 Кб BASIC (позже переросший в 8 Кб BASIC), Extended Basic, Extended ROM BASIC, Disk BASIC.

AmigaBASIC

Реализация языка программирования [BASIC](http://progopedia.ru/language/basic/)

AmigaBASIC — это реализация языка программирования BASIC для ПК фирмы Commodore, спроектированная и написанная корпорацией Microsoft. AmigaBASIC поставлялся с версиями AmigaOS 1.1 и 1.3. AmigaBASIC заменил ABasiC, который был включен в AmigaOS 1.0 и 1.1, и в свою очередь был заменен языком сценариев REXX-подобным — ARexx в версии AmigaOS 2.0.

AmigaBASIC обеспечивал не только общий язык BASIC, но также попытался обеспечить удобные в работе API для уникальных графических и звуковых возможностей AmigaOS. Объектные команды облегчали создание двигающихся объектов, спрайты и так далее.

Популярность AmigaBASIC была такова, что популярный компьютерный журнал включал много программ написанных на AmigaBASIC в статьи, и любой читатель мог использовать данную программу в своем ПК фирмы Commodore. Листинги исходного текста обычно были реализациями простых программ, таких как элементарные игры, аналоговые часы, адресные книги и так далее.

Applesoft BASIC Interpreter in Javascript — это интерпретатор диалекта [Applesoft BASIC](http://progopedia.ru/dialect/applesoft-basic/) реализованный на Javascript.

Atari Microsoft BASIC и Atari Microsoft BASIC II — это реализации языка программирования BASIC диалекта [Microsoft BASIC](http://progopedia.ru/dialect/microsoft-basic/) для 8-битовых машин Atari.

Компания Atari изначально лицензировала Microsoft BASIC для использования в своих 8-битовых компьютерах, но они не смогли приспособить его для работы с ROM 8 Кб. И они обратились к компании Shepardson Microsystems Inc которая создала Atari BASIC.

Atari Microsoft BASIC, в отличие от Atari BASIC, не позволял использовать сокращения ключевых слов. Ключевые слова должны были быть полностью написаны. Проверка синтаксиса происходила после запуска программы, а не немедленно после ввода строки кода. Помимо этого, арифметические операции с целыми числами приводили к целочисленному результату.

Atari Microsoft BASIC поставлялся в двух пакетах:

* на гибком диске CX8126
* на картридже RX8035

Сам картридж мог содержать 16 Кб информации и остающийся файл в 11 поставлялся на отдельном диске. За кассетной версией закрепилось название Atari Microsoft BASIC II.

Не смотря на более широкие возможности особенности Atari Microsoft BASIC, чем у Atari BASIC, он не пользовался широкой популярностью. И Atari BASIC имел большее распространение.

Самыми весомыми проблемами Atari Microsoft BASIC были:

* повышенные требования памяти (не менее 32 Кб)
* наличие дисковода
* низкая производительность
* отсутствие совместимости с Atari BASIC
* большая стоимость

BASICA (Advanced BASIC) — является простым интерпретатором языка BASIC корпорации Microsoft для PC-DOS. BASICA использует ROM BASIC, который включался в ранние модели IBM PC. Данный интерпретатор не запускается на не IBM PC и на поздних моделях IBM PC из-за отсутствия ROM BASIC.

Среда разработки BASICA весьма похожа на ту, что использовалась в Dartmouth BASIC. В обеих пользователь вводит команды. Если команда начинается с номера строки, она считается частью текущей программы, иначе исполняется немедленно.

Преемником BASICA стал GW-BASIC от Microsoft, который был очень похож, но не использует ROM BASIC и как следствие может исполнятся практически на любой системе, совместимой с IBM.

IBM Rational Ada Developer

Реализация языка программирования [Ada](http://progopedia.ru/language/ada/)

В 1985 г. компанией Rational Machines была выпущена Rational Environment, интегрированная среда разработки для языка [Ada](http://progopedia.ru/language/ada/). Rational Environment выполнялся на особом оборудовании, Rational R1000, которое было оптимизировано для Ады. Устройство R1000 позволяло ему одновременно вычислять и проверять границы. Впоследствии имя компании было изменено с "Rational Machines" на "Rational", чтобы избежать акцентирования на этом оборудовании. Rational предоставляла генераторы кода и кросс–отладчики для популярных на то время архитектур VAX, Motorola 68000, и x86. В 1993 Rational начала разработку Rational Apex, полной перереализации Rational Environment для [Ada](http://progopedia.ru/language/ada/). Apex работал на платформах Sun и IBM. Впоследствии была выпущена версия Apex для Windows NT. 6 декабря 2002 г. было объявлено о поглощении Rational в IBM. К тому времени Rational Apex уже поддерживал Ada95. После поглощения продажей и поддержкой среды разработки и компилятора занимается IBM.

Будущее IBM Rational Ada Developer не известно. С одной стороны, для IBM Ada — не основной источник доходов. С другой стороны, в проверке тестов ACATS для стандарта Ada 2005 участвовали по меньшей мере три компании. К сожалению, политика неразглашения не позволяет назвать их.

Язык программирования Питон

Интерпретируемый объектно-ориентированный язык программирования высокого уровня с динамической типизацией, автоматическим управлением памятью и удобными высокоуровневыми структурами данных, такими как словари (хэш-таблицы), списки, кортежи. Поддерживает классы, модули (которые могут быть объединены в пакеты), обработку исключений, а также многопоточные вычисления. Питон обладает простым и выразительным синтаксисом. Язык поддерживает несколько парадигм программирования: структурное, объектно-ориентированное, функциональное и аспектно-ориентированное.

Язык программирования

Pascal — это императивный язык программирования, разработанный Никлаусом Виртом в 1970 в качестве языка обучения структурному программированию. Название языку дано в честь выдающегося французского математика, физика, литератора и философа Блеза Паскаля. Первоначально язык компилировался в байт-код, подобно языку [Java](http://progopedia.ru/language/java/).

Особенностями языка являются строгая типизация и наличие средств структурного (процедурного) программирования. Паскаль был одним из первых таких языков. По мнению Н. Вирта, язык должен способствовать дисциплинированию программирования, поэтому, наряду со строгой типизацией, в Паскале сведены к минимуму возможные синтаксические неоднозначности, а сам синтаксис интуитивно понятен даже при первом знакомстве с языком.

Тем не менее, первоначально язык обладал множеством недостатков: невозможность передачи функциям массивов переменной длины, отсутствие нормальных средств работы с динамической памятью, ограниченная библиотека ввода-вывода, отсутствие средств для подключения функций написанных на других языках, отсутствие средств раздельной компиляции и т. п.

Диалект Object Pascal поддерживает парадигму объектно-ориентированного программирования.

SQL - язык управления базами данных для реляционных баз данных. Сам по себе SQL не является языком программирования, но его стандарт позволяет создавать для него процедурные расширения, которые расширяют его функциональность до полноценного языка программирования.

Язык был создан в 1970х годах под названием "SEQUEL" для системы управления базами данных (СУБД) System R. Позднее он был переименован в "SQL" во избежание конфликта торговых марок. В 1979 году SQL был впервые опубликован в виде коммерческого продукта Oracle V2.

Первый официальный стандарт языка был принят ANSI в 1986 году и ISO - в 1987. С тех пор были созданы еще несколько версий стандарта, некоторые из них повторяли предыдущие с незначительными вариациями, другие принимали новые существенные черты.

Несмотря на существование стандартов, большинство распространенных реализаций SQL отличаются так сильно, что код редко может быть перенесен из одной СУБД в другую без внесения существенных изменений. Это объясняется большим объемом и сложностью стандарта, а также нехваткой в нем спецификаций в некоторых важных областях реализации.

SQL создавался как простой стандартизированный способ извлечения и управления данными, содержащимися в реляционной базе данных. Позднее он стал сложнее, чем задумывался, и превратился в инструмент разработчика, а не конечного пользователя. В настоящее время SQL (по большей части в реализации Oracle) остается самым популярным из языков управления базами данных, хотя и существует ряд альтернатив.

SQL состоит из четырех отдельных частей:

1. язык определения данных (DDL) используется для определения структур данных, хранящихся в базе данных. Операторы DDL позволяют создавать, изменять и удалять отдельные объекты в БД. Допустимые типы объектов зависят от используемой СУБД и обычно включают базы данных, пользователей, таблицы и ряд более мелких вспомогательных объектов, например, роли и индексы.
2. язык манипуляции данными (DML) используется для извлечения и изменения данных в БД. Операторы DML позволяют извлекать, вставлять, изменять и удалять данные в таблицах. Иногда операторы select извлечения данных не рассматриваются как часть DML, поскольку они не изменяют состояние данных. Все операторы DML носят декларативный характер.
3. язык определения доступа к данным (DCL) используется для контроля доступа к данным в БД. Операторы DCL применяются к привилегиям и позволяют выдавать и отбирать права на применение определенных операторов DDL и DML к определенным объектам БД.
4. язык управления транзакциями (TCL) используется для контроля обработки транзакций в БД. Обычно операторы TCL включают commit для подтверждения изменений, сделанных в ходе транзакции, rollback для их отмены и savepoint для разбиения транзакции на несколько меньших частей.

Следует отметить, что SQL реализует декларативную парадигму программирования: каждый оператор описывает только необходимое действие, а СУБД принимает решение о том, как его выполнить, т.е. планирует элементарные операции, необходимые для выполнения действия и выполняет их. Тем не менее, для эффективного использования возможностей SQL разработчику необходимо понимать то, как СУБД анализирует каждый оператор и создает его план выполнения.

SQL - язык управления базами данных для реляционных баз данных. Сам по себе SQL не является языком программирования, но его стандарт позволяет создавать для него процедурные расширения, которые расширяют его функциональность до полноценного языка программирования.

Язык был создан в 1970х годах под названием "SEQUEL" для системы управления базами данных (СУБД) System R. Позднее он был переименован в "SQL" во избежание конфликта торговых марок. В 1979 году SQL был впервые опубликован в виде коммерческого продукта Oracle V2.

Первый официальный стандарт языка был принят ANSI в 1986 году и ISO - в 1987. С тех пор были созданы еще несколько версий стандарта, некоторые из них повторяли предыдущие с незначительными вариациями, другие принимали новые существенные черты.

Несмотря на существование стандартов, большинство распространенных реализаций SQL отличаются так сильно, что код редко может быть перенесен из одной СУБД в другую без внесения существенных изменений. Это объясняется большим объемом и сложностью стандарта, а также нехваткой в нем спецификаций в некоторых важных областях реализации.

SQL создавался как простой стандартизированный способ извлечения и управления данными, содержащимися в реляционной базе данных. Позднее он стал сложнее, чем задумывался, и превратился в инструмент разработчика, а не конечного пользователя. В настоящее время SQL (по большей части в реализации Oracle) остается самым популярным из языков управления базами данных, хотя и существует ряд альтернатив.

SQL состоит из четырех отдельных частей:

1. язык определения данных (DDL) используется для определения структур данных, хранящихся в базе данных. Операторы DDL позволяют создавать, изменять и удалять отдельные объекты в БД. Допустимые типы объектов зависят от используемой СУБД и обычно включают базы данных, пользователей, таблицы и ряд более мелких вспомогательных объектов, например, роли и индексы.
2. язык манипуляции данными (DML) используется для извлечения и изменения данных в БД. Операторы DML позволяют извлекать, вставлять, изменять и удалять данные в таблицах. Иногда операторы select извлечения данных не рассматриваются как часть DML, поскольку они не изменяют состояние данных. Все операторы DML носят декларативный характер.
3. язык определения доступа к данным (DCL) используется для контроля доступа к данным в БД. Операторы DCL применяются к привилегиям и позволяют выдавать и отбирать права на применение определенных операторов DDL и DML к определенным объектам БД.
4. язык управления транзакциями (TCL) используется для контроля обработки транзакций в БД. Обычно операторы TCL включают commit для подтверждения изменений, сделанных в ходе транзакции, rollback для их отмены и savepoint для разбиения транзакции на несколько меньших частей.

Следует отметить, что SQL реализует декларативную парадигму программирования: каждый оператор описывает только необходимое действие, а СУБД принимает решение о том, как его выполнить, т.е. планирует элементарные операции, необходимые для выполнения действия и выполняет их. Тем не менее, для эффективного использования возможностей SQL разработчику необходимо понимать то, как СУБД анализирует каждый оператор и создает его план выполнения.

BARSIC (Business And Research Scientific Interactive Calculator) является интерпретируемым языком программирования с псевдокомпиляцией исходного кода в brc-файл (сокращение от BARSIC Compiled). Первоначально возник как надстройка над библиотеками управления научными и учебными установками. В том числе со средствами отображения графиков и проведения вычислений по формулам, задаваемым пользователем. В дальнейшем были добавлены средства визуального проектирования пользовательского интерфейса. Это объектный язык с развитыми средствами процедурного программирования (функции-выражения, подпрограммы с различными модификаторами видимости внешних элементов внутри подпрограмм, модули, структуры) и элементами объектности (набор встроенных предопределённых классов как способ организации удобного доступа к библиотекам). Предполагается добавление пользовательских классов по модели прототипирования.

Среди интерпретируемых языков отличается наличием жёсткой статической проверки типов во всех синтаксических конструкциях за исключением нескольких, специально предназначенных для интерпретации. Динамическая проверка также ведётся. При этом синтаксические конструкции языка выбраны таким образом, чтобы транслятору удалось максимально точно локализовать место синтаксической ошибки. Благодаря этому резко повышается скорость разработки программ и нахождения ошибок.

В настоящее время наиболее широко используемыми приложениями BARSIC являются учебные модели по физике и математике:

* доступная в открытом доступе виртуальная лаборатория [barsic.spbu.ru](http://barsic.spbu.ru/www/lab1108/index.html) (первоначально необходимо инсталлировать на компьютере среду BARSIC [с данной странички](http://barsic.spbu.ru/www/download_r.html)
* [модели и тесты Интернет-олимпиады по физике по СПб и Северо-Западному региону РФ](http://barsic.spbu.ru/olymp/index.html) . К настоящему времени через Интернет-олимпиады по физике на основе тестов и моделей, выполняемых в среде BARSIC, прошло около 5000 человек (только в ноябре 2007 года 1083 человек по 10-м классам и 1246 человек по 11-м классам).

### Состав программного комплекса BARSIC

* бесплатная свободно распространяемая исполняющая среда barsic.exe (и сопровождающие файлы) - 'проигрыватель' файлов BARSIC
* среда barsicIDE.exe разработки программ-приложений, свободно доступная для некоммерческого использования

### Основные особенности языка и среды BARSIC

Программный комплекс BARSIC предназначен для разработки и использования прикладных программ-приложений, работающих под управлением исполняющей среды BARSIC. Основными областями применения комплекса являются разработка программ учебного назначения в области физики, математики, информатики и других областях, а также программ для научных численных математических расчетов и математического моделирования, преимущественно в области физики. Кроме того, комплекс специализирован для создания программ интерактивного управления компьютеризированными экспериментальными установками на основе персональных компьютеров, а также обработки и визуализации получаемых экспериментальных данных и сравнения их с теоретическими зависимостями.

Язык BARSIC, с помощью которого осуществляется программирование программ-приложений, является универсальным объектным языком программирования и содержит развитые средства построения двумерных и трехмерных графиков, анимации изображений, математической обработки данных, управления экспериментом. Среда разработки содержит развитые средства визуального проектирования интерфейса приложений.

Относительная независимость приложений от конкретной версии исполняющей среды дает важное преимущество: добавление новых возможностей в среде автоматически расширяет возможности программы-приложения. Например, работа с файлами, графиками, базами данных, компьютерными сетями и т.д. Самой важной особенностью языка BARSIC является надежность программирования. В BARSIC предусмотрены специальные средства повышения надежности.

Во-первых, выбран такой синтаксис языка программирования, чтобы из-за случайных опечаток во время набора программы минимизировать вероятность возникновения недиагностируемых ошибок. В отличие от BARSIC, такие языки, как [FORTRAN](http://progopedia.ru/language/fortran/), JavaScript, [BASIC](http://progopedia.ru/language/basic/) в очень сильной степени подвержены этим ошибкам из-за правил автоматического приведения типов, а также свободной типизации (для JavaScript и Visual BASIC). В несколько меньшей степени эти ошибки характерны для [C](http://progopedia.ru/language/c/) и [C++](http://progopedia.ru/language/c-plus-plus/), а в еще меньшей степени – для [Java](http://progopedia.ru/language/java/) (за исключением работы со строками, где такие ошибки для [Java](http://progopedia.ru/language/java/) типичны) и [Object PASCAL](http://progopedia.ru/dialect/object-pascal/) (Delphi).

Во-вторых, синтаксис языка программирования BARSIC построен на принципе максимальной понятности для человека и принципе автодокументирования программного кода. Это означает, что сам текст программы в подавляющем большинстве случаев дает возможность понять, что имел в виду программист. Поэтому в программах, написанных на BARSIC, почти не надо вставлять комментарии, что не только экономит время, но и избавляет от неприятной нетворческой работы, которую так не любит большинство программистов. Во всех других известных нам языках программирования этот принцип не выполняется.

В-третьих, в BARSIC на уровне среды встроена система перехвата и обработки так называемых исключительных ситуаций. При этом если программист заранее не предусмотрел действий при возникновении деления на ноль, ввода вместо числа другого символа и т.п., программа не завершит работу в аварийном режиме, как бывает в большинстве программ, написанных на других языках программирования. Среда BARSIC выдаст диагностику ошибки, после чего можно будет повторить ввод с исправлением неправильно введенных значений или символов и продолжить работу. Обработка исключительных ситуаций имеется в большинстве современных языков программирования, но BARSIC обеспечивает дополнительную защиту от исключений на уровне исполняющей среды даже в тех случаях, когда программы, написанные на [C++](http://progopedia.ru/language/c-plus-plus/) или [Object PASCAL](http://progopedia.ru/dialect/object-pascal/), окажутся неработоспособны.

В-четвертых, в интегрированную среду BARSIC встроена поддержка работы с пользовательским интерфейсом методом визуального проектирования. Это делает BARSIC средой RAD (Rapid Application Development) —средством сверхбыстрой разработки программ.

### Языки программирования и интегрированные среды

По словам создателя первой интегрированной среды FRAMEWORK, интегрированная среда —это такая прикладная программа, что пользователь, запустив ее в начале рабочего дня, находит в ней все необходимые для работы ресурсы и поэтому не выходит из интегрированной среды до самого конца рабочего дня. Конечно, это определение не очень корректно и несколько идеализирует ситуацию, но его общий смысл достаточно ясен.

Основная особенность интегрированных сред —высокая степень интерактивности. Она достигается за счет интеграции в единое целое различных программных ресурсов, отсюда и происходит название. Так, интегрированная среда какого-либо компилятора языка программирования (программы, которая из текста данного языка программирования создает исполняемую программу) обычно содержит текстовый редактор и собственно компилятор с системой диагностики ошибок компиляции. Кроме того, в ней обычно имеется также отладчик —интерпретатор данного языка, выполняющий программу строчка за строчкой и имеющий ряд других специальных возможностей.

Одно из активно развивающихся направлений, визуальное проектирование —полностью основано на использовании возможностей интегрированной среды. Пользователь в интерактивном режиме выбирает необходимые для его программы объекты языка программирования и устанавливает между ними связи. Популярность таких языков как [Visual BASIC](http://progopedia.ru/dialect/visual-basic/) (Microsoft), а также [Object PASCAL](http://progopedia.ru/dialect/object-pascal/) (среды Delphi и Kylix, Borland), не случайна. Даже неопытный программист, не знающий кроме [BASIC](http://progopedia.ru/language/basic/) других языков программирования и никогда не программировавший под Windows, может за два-три дня с помощью [Visual BASIC](http://progopedia.ru/dialect/visual-basic/) создать прикладную программу, работающую под Windows. А вот программисту высокого класса, не программировавшему до того под Windows, с помощью [C++](http://progopedia.ru/language/c-plus-plus/) зачастую приходится для создания такой же программы затратить недели, а то и месяцы. Правда, [Visual BASIC](http://progopedia.ru/dialect/visual-basic/) обладает рядом существенных ограничений. С помощью сред визуального проектирования можно создавать весьма сложные программы, не набрав с клавиатуры ни строчки кода.

Однако у всех программ, созданных на основе традиционных языков программирования процедурного типа, имеется один и тот же недостаток. Для них исполняемый код —это одно, а обрабатываемые программой данные —совсем другое. Действительно, код программы содержится в файле с расширением EXE, а данные —либо в специальных файлах данных (как правило, в текстовом либо двоичном виде во внутреннем представлении компьютера), либо вводятся с клавиатуры или с какого либо другого внешнего устройства.

А теперь зададим вопрос: как быть, если пользователь должен дать исполняемой программе информацию, которую можно рассматривать как "добавку" к тексту программы? Например, мы хотим, чтобы на экране был построен график функции, и в подобной программе обеспечиваем все необходимые сервисные возможности. Однако формулу для функции должен задать сам пользователь, и заранее неизвестно, какая она будет.

Совершенно очевидно, что подобного рода задачи можно решать только с помощью системы-интерпретатора. Но "за все приходится платить". Компилятор переводит текст программы в исполняемый код, который может работать и без программы-компилятора. Программы же, созданные на основе языков интерпретирующего типа, могут исполняться только под управлением программы-интерпретатора. Кроме того, они работают медленнее скомпилированных, так как интерпретация занимает дополнительное время. Однако во многих случаях это несущественно.

Cyclone — это типобезопасный аналог языка [C](http://progopedia.ru/language/c/). По утверждению создателей, портирование существующей программы на Cyclone не должно занимать много времени.

В языке Cyclone сделан большой упор на управление памятью. Поддерживаются разные схемы: регионы, счётчики ссылок, сборка мусора.

В процессе развития язык приобрёл дополнительные полезные возможности, такие как кортежи, контейнеры, pattern matching.

ECMAScript — это скриптовый язык программирования, стандартизированный организацией Ecma International в спецификации ECMA-262.

Язык широко используется в интернете. Часто фигурирует под названием JavaScript или JScript (названия двух основных диалектов языка).