Языки программирования

Муллахметова Алина
11-а

Ассемблер

На протяжении 1950-х годов запросы на разработку программного обеспечения возросли и программы стали очень большими. Приходилось писать очень много кода, хотя обеспечение и было весьма простым: по тем временам дизайн рабочего стола был проще нынешнего, программы работали с элементарными вещами, а компьютер только ещё начинал победно шествовать. Однако программы запутывались всё больше, их структура усложнилась, потому что всё время развивалась компьютерная техника. Тогда стали пользоваться специальными программами-сборщиками программ из маленьких кусочков кодов — ассемблерами. Начался новый этап развития.

Теперь, когда была нужна эффективная программа, вместо машинных языков использовались близкие к ним машинно-ориентированные языки ассемблера. К таковым относились, например, Autocode, с 1954-го г. — IPL (предшественник языка LISP), с 1955-го г. — FLOW-MATIC . Теперь люди стали использовать мнемонические команды взамен машинных команд.

Но даже работа с ассемблером достаточно сложна и требует специальной подготовки. Например, для процессора Zilog Z80 машинная команда 00000101 предписывает процессору уменьшить на единицу свой регистр B. На языке ассемблера это же будет записано как DEC B.

BASIC

Бейсик (BASIC - Beginner's All-Purpose Symbolic Instruction Code – "универсальный символический код инструкций для начинающих"). Прямой потомок Фортрана и до сих пор самый популярный язык программирования для персональных компьютеров. Появился Бейсик в 1963 году (назвать автора было бы трудно, но основная заслуга в его появлении несомненно принадлежит американцам Джону Кемени и Томасу Курцу). Как и любые преимущества, простота Бейсика оборачивалась, особенно в ранних версиях трудностями структурирования; кроме того, Бейсик не допускал рекурсию – интересный прием, позволяющий составлять эффективные и в то же время короткие программы.

Разработаны мощные компиляторы Бейсика, которые обеспечивают не только богатую лексику и высокое быстродействие, но и возможность структурного программирования. По мнению некоторых программистов, наиболее интересными версиями являются GWBASIC, Turbo-Basic и Quick Basic.

В свое время появление Quick Basic ознаменовало рождение второго поколения систем программирования на языке Бейсик. Он предоставлял возможность модульного и процедурного программирования, создания библиотек, компиляции готовых программ и прочее, что вывело его на уровень таких классических языков программирования, как Си, Паскаль, Фортран и др. Более того, в связи с отсутствием официального стандарта языка Бейсик, его реализация в виде Quick Basic стала фактическим стандартом. Безусловными лидерами среди различных версий Бейсика были Quick Basic 4.5 и PDS 7.1 фирмы Microsoft, появившиеся в конце 80-х годов.

Фортран (Fortran)

Фортран самый первый из языков высокого уровня (разработан Бэкусом в начале 1950-х годов) и широко распространенный язык, особенно среди пользователей, которые занимаются численным моделированием. Это объясняется несколькими причинами:

- существованием огромных фондов прикладных программ на Фортране, накопленных за эти годы, а также наличием огромного количества программистов, эффективно использующих этот язык;

- наличием эффективных трансляторов Фортрана на всех типах ЭВМ, причем версии для различных машин достаточно стандартизированы и перенос программ с машины на машину обычно не составляет больших трудностей;

- изначальной направленностью Фортрана на физико-математические и технические приложения; в частности, это проявилось в том, что в течение долгого времени он оставался единственным языком со встроенным комплексным типом переменных и большим набором встроенных функций для работы с такими переменными.

За прошедший период сформировалась новая методология и философия программирования. С начала 70-х годов Фортран подвергался заслуженной критике. В 1977 году был принят новый стандарт языка Фортран-77. На создание нового стандарта ушло много времени, но сейчас уже можно считать, что его разработка завершена и новый стандарт Фортран-90 начал входить в практику пользователей Фортрана. Только на машинах типа IBM PC существует несколько трансляторов например, Watfor, Lap-Fortran и т. д. Но наибольшее распространение на машинах этого типа получили различные версии транслятор Fortran-77. Выпущенный в 1990 году транслятор MS-Fortran 5.0 практически полностью соответствует стандарту Fortran-90. Большинство крупных научно-технических прикладных программ написано на Фортране потому, что он обладает переносимостью и устойчивостью, а также благодаря наличию встроенных математических и тригонометрических функций. Дополнительной, неотъемлемой частью любой прикладной программы на языке Фортран является расширенная графическая библиотека, позволяющая использовать различные графические данные и изображения.

Кобол

Кобол - это один из первых языков программирования, появившийся в 1959, разработанный прежде всего для исследований в экономической сфере. Язык позволяет эффективно работать с большим количеством данных, он насыщен разнообразными возможностями поиска, сортировки и распределения. О программах на Коболе, основанных на широком использовании английского языка, говорят, что они понятны даже тем, кто не владеет Коболом, поскольку тексты на этом языке программирования не нуждаются в каких-либо специальных комментариях. Подобные программы принято называть самодокументирующимися. К числу других плюсов Кобола обычно относят его структурированность. Довольно мощные компиляторы с этого языка разработаны для персональных компьютеров. Некоторые из них столь эффективны, что программу, отлаженную на персональном компьютере, нетрудно перенести на большие ЭВМ.

Перечисляя минусы нельзя не вспомнить о том, что на Коболе можно запрограммировать лишь простейшие алгебраические вычисления. Для инженерных расчетов этот язык не годится. Еще одна причина, которая в какой-то мере сдерживает развитие языка, - это наличие в США специально созданного отраслевого комитета, вырабатывающего стандарты, за соблюдением которых следит правительственная комиссия. Как это всегда бывает в подобных случаях, фирмы, занимающиеся разработкой программного обеспечения, не торопятся подгонять свои заготовки к жестким требованиям комиссии, отсутствует конкуренция версий, а в итоге проигрывает распространение языка

PASCAL

Язык программирования Паскаль был разработан профессором кафедры вычислительной техники Швейцарского Федерального института технологии Николасом Виртом в 1968 году как альтернатива существующим и все усложняющимся языкам программирования, таким, как PL/1, Algol, Fortran. Интенсивное развитие Паскаля привело к появлению уже в 1973 году его стандарта в виде пересмотренного сообщения, а число трансляторов с этого языка в 1979 году перевалило за 80. В начале 80-х годов Паскаль еще более упрочил свои позиции с появлением трансляторов MS-Pascal и Turbo-Pascal для ПЭВМ. С этого времени Паскаль становится одним из наиболее важных и широко используемых языков программирования. Существенно то, что язык давно вышел за рамки академического и узко профессионального интереса и используется в большинстве университетов высокоразвитых стран не только как рабочий инструмент пользователя. Важнейшей особенностью Паскаля является воплощенная идея структурного программирования. Другой существенной особенностью является концепция структуры данных как одного из фундаментальных понятий.

Основные причины популярности Паскаля заключаются в следующем:

- простота языка позволяет быстро его освоить и создавать алгоритмически сложные программы

- развитые средства представления структур данных обеспечивают удобство работы как с числовой, так и с символьной и битовой информацией

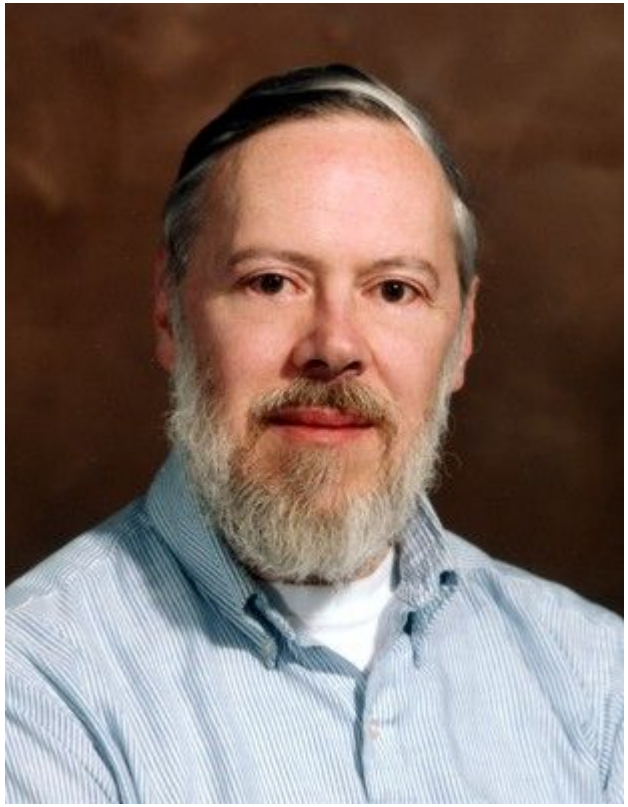
- наличие специальных методик создания трансляторов с Паскаля упростило их разработку и способствовало широкому распространению языка

- оптимизирующие свойства трансляторов с Паскаля позволяют создавать эффективные программы. Это послужило одной из причин использования Паскаля в качестве языка системного программирования

- в языке Паскаль реализуются идеи структурного программирования, что делает программу наглядной и дает хорошие возможности для разработки и отладки

C

Сотрудник фирмы Bell Labs Денис Ритчи создал язык Си в 1972 году во время совместной работы с Кеном Томпсоном, как инструментальное средство для реализации операционной системы Unix, однако популярность этого языка быстро переросла рамки конкретной операционной системы и конкретных задач системного программирования. В настоящее время любая инструментальная и операционная система не может считаться полной если в ее состав не входит компилятор языка Си. Ритчи не выдумывал Си просто из головы – прообразом служил язык Би разработанный Томпсоном. Язык программирования Си был разработан как инструмент для программистов-практиков. В соответствии с этим главной целью его автора было создание удобного и полезного во всех отношениях языка.



Си является орудием системного программиста и позволяет глубоко влезать в самые тонкие механизмы обработки информации на ЭВМ. Хотя язык требует от программиста высокой дисциплины, он не строг в формальных претензиях и допускает краткие формулировки.

Си – современный язык. Он включает в себя те управляющие конструкции, которые рекомендованы теорией и практикой программирования. Его структура побуждает программиста использовать в своей работе нисходящее проектирование, структурное программирование и пошаговую разработку модулей.

Си – мощный и гибкий язык. Большая часть операционной системы Unix, компиляторы и интерпретаторы языков Фортран, Паскаль, Лисп, и Бейсик написаны именно с его помощью.

Си – удобный язык. Он достаточно структурирован, чтобы поддерживать хороший стиль программирования и вместе с тем не связан жесткими ограничениями. В некотором смысле язык Си – самый универсальный, т.к. кроме набора средств, присущих современным языкам программирования высокого уровня (структурность, модульность, определенные типы данных), в него включены средства для программирования практически на уровне ассемблера. Большой набор операторов и средств требуют от программиста осторожности, аккуратности и хорошего знания языка со всеми иго преимуществами и недостатками.

C++

Язык C++ появился в начале 80-х годов. Созданный Бьерном Страуструпом с первоначальной целью избавить себя и своих друзей от программирования на ассемблере, Си или различных других языках высокого уровня.

По мнению автора языка, различие между идеологией Си и C++ заключается примерно в следующем: программа на Си отражает “способ мышления” процессора, а C++ - способ мышления программиста. Отвечая требованиям современного программирования, C++ делает акцент на разработке новых типов данных наиболее полно соответствующих концепциям выбранной области знаний и задачам приложения. Класс является ключевым понятием C++. Описание класса содержит описание данных, требующихся для представления объектов этого типа и набор операций для работы с подобными объектами.

В отличие от традиционных структур Си и Паскаля, членами класса являются не только данные, но и функции. Функции – члены класса имеют привилегированный доступ к данным внутри объектов этого класса и обеспечивают интерфейс между этими объектами и остальной программой. При дальнейшей работе совершенно не обязательно помнить о внутренней структуре класса и механизме работы встроенных функций. В этом смысле класс подобен электрическому прибору – мало кто знает о его устройстве, но все знают, как им пользоваться.

Язык C++ является средством объектного программирования, новейшей методики проектирования и реализации программ, которая в текущем десятилетии, скорее всего, заменит традиционное процедурное программирование. Главной целью создателя языка доктора Бьерна Страустрапа было оснащение языка C++ конструкциями, позволяющими увеличить производительность труда программистов и облегчить процесс овладения большими программными продуктами.

Абстракция, реализация, наследование и полиморфизм являются необходимыми свойствами которыми обладает язык C++, благодаря чему он не только универсален, как и язык Си, но и является объектным языком.

Ада

Язык Ада создан в основном в 1975 - 1980 годах в результате грандиозного проекта, предпринятого Министерством Обороны США с целью разработать единый язык программирования для так называемых встроенных систем (т. е. систем управления автоматизированными комплексами, работающими в реальном времени). Имелись в виду прежде всего бортовые системы управления военными объектами (кораблями, самолетами, танками, ракетами, снарядами и т. п.). Поэтому решения, принятые авторами Ады не следует считать универсальными. Их нужно воспринимать в контексте особенностей выбранной предметной области. Язык Ада возник в результате международного конкурса языковых проектов проходящего в 1978-1979 годах. Участники должны были удовлетворить довольно жестким, детально разработанным под эгидой Министерства Обороны США требованиям. Интересно, что все языки, дошедшие до последних туров этого конкурса, были основаны на Паскале. В этой связи Аду можно предварительно охарактеризовать как Паскаль, развитый с учетом перечисленных выше пяти основных требований. При этом авторы пошли в основном по пути расширения Паскаля новыми элементами. В результате получился существенно более сложный язык.

PL/1

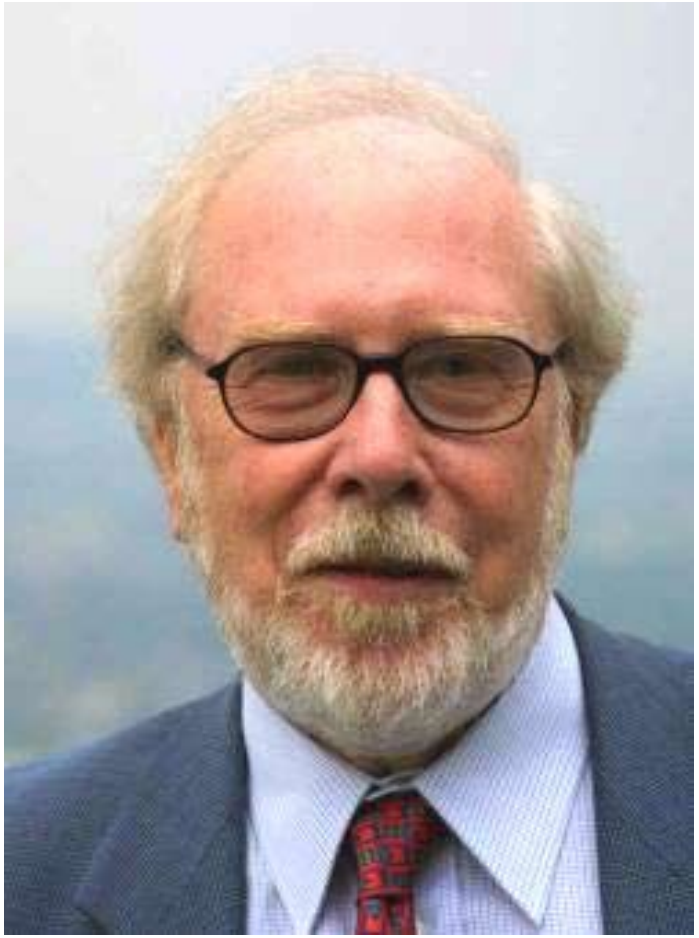
PL/1 разработан в 1964-1965 годах фирмой IBM. PL/1 относится к числу универсальных языков, т. е. позволяет решать задачи из разных областей: численные расчеты, текстовая обработка, экономические задачи и т. д. По своим возможностям он перекрывает такие языки, как Фортран, Алгол-60 (созданный для численных расчетов), Кобол (для экономических задач), хотя в силу ряда причин вытеснить эти языки PL/1 не смог.

PL/1 содержит все основные конструкции, характерные для так называемых языков высокого уровня, а также ряд специфичных средств, удобных для практического программирования. Язык напоминает конструктор с большим числом деталей – пользователю достаточно освоить только те части языка, которые ему практически необходимы. Его операторы довольно емки, что часто позволяет получить запись программы более компактную, чем на других языках. Знающий PL/1 программист без труда осваивает любой другой язык того же или близкого класса.

Вместе с тем, ПЛ/1 имеет и ряд недостатков, затрудняющих изучение и использование языка. Основные из них таковы. Во-первых, имеется много дублирующих друг друга средств их сложно запомнить, не ясно, что когда применять, кроме того, это снижает как скорость трансляции, так и скорость выполнения программ. Во-вторых, программы получаются не совсем машинно-независимыми.

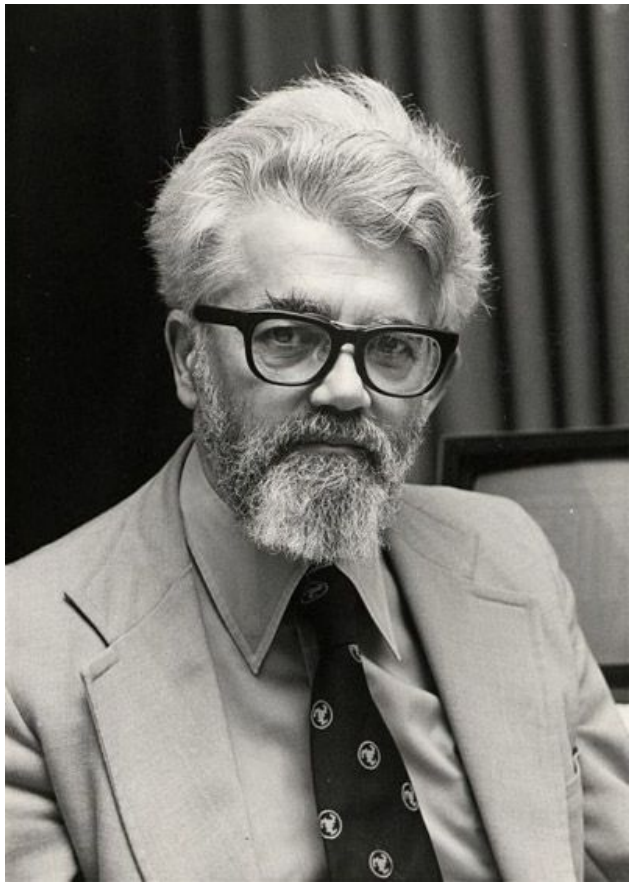
Модула

Можно считать, что история языка Модула начинается в 1980 году, когда Никлаус Вирт, один из выдающихся специалистов по теории информации, известный большинству специалистов по вычислительной технике в основном как создатель языка Паскаль, опубликовал описание нового языка программирования, названного им Модула. В отличие от Паскаля, который был по замыслу языком для обучения программирования, Модула с самого начала представлял собой язык для профессиональных системных программистов, продолжая лучшие традиции своего предшественника и обогащая их новыми идеями, соответствующих таким требованиям к языкам программирования, как структурность, модульность и способность к расширению. Как и множество других языков программирования, Модула подвергалась эволюции, во время которой ее первоначальное название было переделано в имя Модула-2. Одновременно с развитием языка Модула для него создавались новые компиляторы, однако, ни один из них не мог соперничать с лучшими реализациями языков Паскаль и Си, например, разработанных фирмой Борланд. В этот переходный для языка Модула период лучшей считались реализации выполненные фирмой Logitech, которые по своим характеристикам проигрывала Турбо Паскалю и Турбо Си. Только в 1988 году после появления на американском рынке системы Top Speed, Модула-2 заняла достойное место среди процедурных языков, предназначенных для системного программирования. Возраставшей популярности системы Top Speed способствовало несколько факторов: удобное и, кроме того, легко изменяемая по желанию пользователей операционное окружение, быстрый компилятор и селективный редактор связей. Но наиболее существенным оказалось то, что создаваемые программы отличались большим быстродействием и занимали не много места в памяти.



Симула

ЛИСП



Язык Лисп был предложен Дж. Маккарти в работе в 1960 году и ориентирован на разработку программ для решения задач не численного характера. Английское название этого языка – LISP является аббревиатурой выражения LISt Processing (обработка списков) и хорошо подчеркивает основную область его применения. Понятие “список” оказалось очень емким. В виде списков удобно представлять алгебраические выражения, графы, элементы конечных групп, множества, правила вывода и многие другие сложные объекты. Списки являются наиболее гибкой формой представления информации в памяти компьютеров. Неудивительно поэтому, что удобный язык, специально предназначенный для обработки списков, быстро завоевал популярность.

После появления Лиспа различными авторами был предложен целый ряд других алгоритмических языков ориентированных на решение задач в области искусственного интеллекта, среди которых можно отметить Плэнер, Снобол, Рефал, Пролог. Однако это не помешало Лиспу остаться наиболее популярным языком для решения таких задач. На протяжении почти сорокалетней истории его существования появился ряд диалектов этого языка: Common LISP, Mac LISP, Inter LISP, Standard LISP и др. Различия между ними не носят принципиального характера и в основном сводятся к несколько отличающемуся набору встроенных функций и некоторой разнице в форме записи программ. Поэтому программист, научившийся работать на одном из них без труда сможет освоить и любой другой. Большим достоинством Лиспа является его функциональная направленность, т. е. программирование ведется с помощью функций. Причем функция понимается как правило, сопоставляющее элементам некоторого класса соответствующие элементы другого класса. Сам процесс сопоставления не оказывает никакого влияния на работу программы, важен только его результат – значение функции. Это позволяет относительно легко писать и отлаживать большие программные комплексы. Ясность программ, четкое разграничение их функций, отсутствие каверзных побочных эффектов при их выполнении является обязательными требованиями к программированию таких логически сложных задач, каковыми являются задачи искусственного интеллекта. Дисциплина в программировании становится особенно важной, когда над программой работает не один человек, а целая группа программистов.

Язык программирования Лисп предназначен в первую очередь для обработки символьной информации. Поэтому естественно, что в мире Лиспа числа играют далеко не главную роль. Основные типы данных в Лиспе называются “атом” и “точечная пара”.

Пролог

Созданный Аланом Колмерауэром Язык логического программирования предназначен для представления и использования знаний о некоторой предметной области. Программы на этом языке состоят из некоторого множества отношений, а ее выполнение сводится к выводу нового отношения на основе заданных. В Прологе реализован декларативный подход, при котором достаточно описать задачу с помощью правил и утверждений относительно заданных объектов. Если это описание является достаточно точным, то ЭВМ может самостоятельно найти требуемое решение. Язык сосредоточен вокруг небольшого набора основных механизмов, включая сопоставление с образцом, древовидного представления структур данных и автоматического перебора с возвратами. Хорошо подходит для решения задач, где рассматриваются объекты (в частности структурированные объекты) и отношения между ними. Пролог, благодаря своим особенностям, используется в области искусственного интеллекта, компьютерной лингвистики и нечислового программирования в целом. В некоторых случаях реализация символьных вычислений на других стандартных языках вызывает необходимость создавать большое количество кода, сложного в понимании, в то время как реализация тех же алгоритмов на языке Пролог дает простую программу, легко помещающуюся на одной странице.

Prolog является декларативным языком программирования: логика программы выражается в терминах отношений, представленных в виде фактов и правил. Для того, чтобы инициировать вычисления, выполняется специальный запрос к базе знаний, на которые система логического программирования генерирует ответы «истина» и «ложь». Для обобщённых запросов с переменными в качестве аргументов созданная система Пролог выводит конкретные данные в подтверждение истинности обобщённых сведений и правил вывода.

Object PAL

Object PAL является мощным языком программирования. Object PAL представляет собой объектно-ориентированный, управляемый по событиям, визуальный язык программирования. На начальном уровне функциональности Object PAL можно осуществлять операции с данными, создавать специальные меню, а также управлять сеансом ввода данных. События в Object PAL порождают команды, которые имитируют эффект использования Paradox в интерактивном режиме. Существует возможность автоматизировать часто выполняемые задания, а также осуществлять над таблицами, формами и отчетами действия, которые были не доступны при интерактивной работе. Также Object PAL предоставляет все средства полнофункционального языка программирования в среде Windows. Можно использовать Object PAL для создания законченных систем, в которых реализованы специальная система меню, справочная система, а также всевозможные проверки данных. В Object PAL можно сохранить свои наработки в динамически компонуемой библиотеке, доступ к которой будут иметь несколько форм. Кроме того, можно установить связь с другими динамическими библиотеками, содержащие прграммы написанные на таких языках как Си, C++ или Паскаль.

Object PAL может быть использован в качестве инструмента для создания автономных программ. Можно написать законченное Windows–приложение и запустить его под Paradox.

Object PAL поддерживает механизм динамического обмена данными в качестве как клиента, так и сервера. Кроме того, Object PAL поддерживает в качестве клиента механизм работы с составными документами. В дополнение к сказанному существует возможность включать в свое приложение мультимедийные средства, снабдив выполняемое приложение звуковыми и анимационными эффектами.

DBase

Java