**ВВЕДЕНИЕ**

Реализация программного обеспечения связана с использованием одного из языков программирования. Показано, что наиболее удобными для реализации программных систем, разработанных в рамках объектно-ориентированного подхода, являются объектно-ориентированные языки программирования, хотя возможна реализация и на обычных (не объектно-ориентированных) языках.

Объектно-ориентированные языки программирования пользуются в последнее время большой популярностью среди программистов, так как они позволяют использовать преимущества объектно-ориентированного подхода не только на этапах проектирования и конструирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

Первый объектно-ориентированный язык программирования Simula 67 был разработан в конце 60-х годов в Норвегии. Авторы этого языка очень точно угадали перспективы развития программирования: их язык намного опередил свое время.

Однако современники (программисты 60-х годов) оказались не готовы воспринять ценности языка Simula 67, и он не выдержал конкуренции с другими языками программирования (прежде всего, с языком Fortran). Прохладному отношению к языку Simula 67 способствовало и то обстоятельство, что он был реализован как интерпретируемый (а не компилируемый) язык, что было совершенно неприемлемым в 60-е годы, так как интерпретация связана со снижением эффективности (скорости выполнения) программ.

Но достоинства языка Simula 67 были замечены некоторыми программистами, и в 70-е годы было разработано большое число экспериментальных объектно-ориентированных языков программирования: например, языки CLU, Alphard, Concurrent Pascal и др. Эти языки так и остались экспериментальными, но в результате их исследования были разработаны современные объектно-ориентированные языки программирования: C++, Smalltalk, Eiffel и др.

Наиболее распространенным объектно-ориентированным языком программирования безусловно является C++. Свободно распространяемые коммерческие системы программирования C++ существуют практически на любой платформе. Широко известна свободно распространяемая система программирования G++, которая дает возможность всем желающим разобрать достаточно хорошо и подробно прокомментированный исходный текст одного из образцовых компиляторов языка C++. Завершается работа по стандартизации языка C++: последний Draft стандарта C++ выпущен в июне 1995 г. (он доступен по Internet).

Изм.

Лист

№ докум.

Подпись

Дата

Лист

Разработка новых объектно-ориентированных языков программирования продолжается. С 1995 года стал широко распространяться новый объектно-ориентированный язык программирования Java, ориентированный на сети компьютеров и, прежде всего, на Internet. Синтаксис этого языка напоминает синтаксис языка C++, однако эти языки имеют мало общего. Java интерпретируемый язык: для него определены внутреннее представление (bytecode) и интерпретатор этого представления, которые уже сейчас реализованы на большинстве платформ. Интерпретатор упрощает отладку программ, написанных на языке Java, обеспечивает их переносимость на новые платформы и адаптируемость к новым окружениям. Он позволяет исключить влияние программ, написанных на языке Java, на другие программы и файлы, имеющиеся на новой платформе, и тем самым обеспечить безопасность при выполнении этих программ. Эти свойства языка Java позволяют использовать его как основной язык программирования для программ, распространяемых по сетям (в частности, по сети Internet).

**1 ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ**

**1.1 ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ**

Технология программирования - это совокупность методов и средств разработки (написания) программ и порядок применения этих методов и средств.

На ранних этапах развития программирования, когда программы писались в виде последовательностей машинных команд, какая-либо технология программирования отсутствовала. Первые шаги в разработке технологии состояли в представлении программы в виде последовательности операторов. Написанию последовательности машинных команд предшествовало составление операторной схемы, отражающей последовательность операторов и переходы между ними. Операторный подход позволил разработать первые программы для автоматизации составления программ - так называемые составляющие программы.

С увеличением размеров программ стали выделять их обособленные части и оформлять их как подпрограммы. Часть таких подпрограмм объединялась в библиотеки, из которых подпрограммы можно было включать в рабочие программы и затем вызывать из рабочих программ. Это положило начало процедурному программированию - большая программа представлялась совокупностью процедур-подпрограмм. Одна из подпрограмм являлась главной и с нее начиналось выполнение программы.

В 1958 году были разработаны первые языки программирования, Фортран и Алгол-58. Программа на Фортране состояла из главной программы и некоторого количества процедур - подпрограмм и функций. Программа на Алголе-58 и его последующей версии Алголе-60представляла собой единое целое, но имела блочную структуру, включающую главный блок и вложенные блоки подпрограмм и функций. Компиляторы для Фортрана обеспечивали раздельную трансляцию процедур и последующее их объединение в рабочую программу, первые компиляторы для Алгола предполагали, что транслируется сразу вся программа, раздельная трансляция процедур не обеспечивалась.

Процедурный подход потребовал структурирования будущей программы, разделения ее на отдельные процедуры. При разработке отдельной процедуры о других процедурах требовалось знать только их назначение и способ вызова. Появилась возможность перерабатывать отдельные процедуры, не затрагивая остальной части программы, сокращая при этом затраты труда и машинного времени на разработку и модернизацию программ.

Следующим шагом в углублении структурирования программ стало так называемое структурное программирование, при котором программа в целом и отдельные процедуры рассматривались как последовательности канонических структур: линейных участков, циклов и разветвлений. Появилась возможность читать и проверять программу как последовательный текст, что повысило производительность труда программистов при разработке и отладке программ. С целью повышения структурности программы были выдвинуты требования к большей независимости подпрограмм, подпрограммы должны связываться с вызывающими их программами только путем передачи им аргументов, использование в подпрограммах переменных, принадлежащих другим процедурам или главной программе, стало считаться нежелательным.

Процедурное и структурное программирование затронули прежде всего процесс описания алгоритма как последовательности шагов, ведущих от варьируемых исходных данных к искомому результату. Для решения специальных задач стали разрабатываться языки программирования, ориентированные на конкретный класс задач: на системы управления базами данных, имитационное моделирование и т.д. При разработке трансляторов все больше внимания стало уделяться обнаружению ошибок в исходных текстах программ, обеспечивая этим сокращение затрат времени на отладку программ.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

Применение программ в самых разных областях человеческой деятельности привело к необходимости повышения надежности всего программного обеспечения. Одним из направлений совершенствования языков программирования стало повышения уровня типизации данных. Теория типов данных исходит из того, что каждое используемое в программе данное принадлежит одному и только одному типу данных. Тип данного определяет множество возможных значений данного и набор операций, допустимых над этим данным. Данное конкретного типа в ряде случаев может быть преобразовано в данное другого типа, но такое преобразование должно быть явно представлено в программе. В зависимости от степени выполнения перечисленных требований можно говорить об уровне типизации того или иного языка программирования. Стремление повысить уровень типизации языка программирования привело к появлению языка Паскаль, который считается строго типизированным языком, хотя и в нем разрешены некоторые неявные преобразования типов, например, целого в вещественное. Применение строго типизированного языка при написании программы позволяет еще при трансляции исходного текста выявить многие ошибки использования данных и этим повысить надежность программы. Вместе с тем строгая типизация сковывала свободу программиста, затрудняла применение некоторых приемов преобразования данных, часто используемых в системном программировании. Практически одновременно с Паскалем был разработан язык Си, в большей степени ориентированный на системное программирование и относящийся к слабо типизированным языкам.

Все универсальные языки программирования, несмотря на различия в синтаксисе и используемых ключевых словах, реализуют одни и те же канонические структуры: операторы присваивания, циклы и разветвления. Во всех современных языках присутствуют предопределенные (базовые) типы данных (целые и вещественные арифметические типы, символьный и, возможно, строковый тип), имеется возможность использования агрегатов данных, в том числе массивов и структур (записей). Для арифметических данных разрешены обычные арифметические операции, для агрегатов данных обычно предусмотрена только операция присваивания и возможность обращения к элементам агрегата. Вместе с тем при разработке программы для решения конкретной прикладной задачи желательна возможно большая концептуальная близость текста программы к описанию задачи. Например, если решение задачи требует выполнения операций над комплексными числами или квадратными матрицами, желательно, чтобы в программе явно присутствовали операторы сложения, вычитания, умножения и деления данных типа комплексного числа, сложения, вычитания, умножения и обращения данных типа квадратной матрицы. Решение этой проблемы возможно несколькими путями:

- Построением языка программирования, содержащего как можно больше типов данных, и выбором для каждого класса задач некоторого подмножества этого языка. Такой язык иногда называют языком-оболочкой. На роль языка-оболочки претендовал язык ПЛ/1, оказавшийся настолько сложным, что так и не удалось построить его формализованное описание. Отсутствие формализованного описания, однако, не помешало широкому применению ПЛ/1 как в Западной Европе, так и в СССР.

- Построением расширяемого языка, содержащего небольшое ядро и допускающего расширение, дополняющее язык типами данных и операторами, отражающими концептуальную сущность конкретного класса задач. Такой язык называют языком-ядром. Как язык-ядро были разработаны языки Симула и Алгол-68, не получившие широкого распространения, но оказавшие большое влияние на разработку других языков программирования.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

Дальнейшим развитием второго пути явился объектно-ориентированный подход к программированию.

**1.2 СУЩНОСТЬ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА К ПРОГРАММИРОВАНИЮ**

Основные идеи объектно-ориентированного подхода опираются на следующие положения:

- Программа представляет собой модель некоторого реального процесса, части реального мира.

- Модель реального мира или его части может быть описана как совокупность взаимодействующих между собой объектов.

- Объект описывается набором параметров, значения которых определяют состояние объекта, и набором операций (действий), которые может выполнять объект.

- Взаимодействие между объектами осуществляется посылкой специальных сообщений от одного объекта к другому. Сообщение, полученное объектом, может потребовать выполнения определенных действий, например, изменения состояния объекта.

- Объекты, описанные одним и тем же набором параметров и способные выполнять один и тот же набор действий представляют собой класс однотипных объектов.

С точки зрения языка программирования класс объектов можно рассматривать как тип данного, а отдельный объект - как данное этого типа. Определение программистом собственных классов объектов для конкретного набора задач должно позволить описывать отдельные задачи в терминах самого класса задач (при соответствующем выборе имен типов и имен объектов, их параметров и выполняемых действий).

Таким образом, объектно-ориентированный подход предполагает, что при разработке программы должны быть определены классы используемых в программе объектов и построены их описания, затем созданы экземпляры необходимых объектов и определено взаимодействие между ними.

Классы объектов часто удобно строить так, чтобы они образовывали иерархическую структуру. Например, класс “Студент”, описывающий абстрактного студента, может служить основой для построения классов “Студент 1 курса”, “Студент 2 курса” и т.д., которые обладают всеми свойствами студента вообще и некоторыми дополнительными свойствами, характеризующими студента конкретного курса. При разработке интерфейса с пользователем программы могут использовать объекты общего класса “Окно” и объекты классов специальных окон, например, окон информационных сообщений, окон ввода данных и т.п. В таких иерархических структурах один класс может рассматриваться как базовый для других, производных от него классов. Объект производного класса обладает всеми свойствами базового класса и некоторыми собственными свойствами, он может реагировать на те же типы сообщений от других объектов, что и объект базового класса и на сообщения, имеющие смысл только для производного класса. Обычно говорят, что объект производного класса наследует все свойства своего базового класса.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

Некоторые параметры объекта могут быть локализованы внутри объекта и недоступны для прямого воздействия извне объекта. Например, во время движения объекта-автомобиля объект-водитель может воздействовать только на ограниченный набор органов управления (рулевое колесо, педали газа, сцепления и тормоза, рычаг переключения передач) и ему недоступен целый ряд параметров, характеризующих состояние двигателя и автомобиля в целом.

Очевидно, для того, чтобы продуктивно применять объектный подход для разработки программ, необходимы языки программирования, поддерживающие этот подход, т.е. позволяющие строить описание классов объектов, образовывать данные объектных типов, выполнять операции над объектами. Одним из первых таких языков стал язык SmallTalk в котором все данные являются объектами некоторых классов, а общая система классов строится как иерархическая структура на основе предопределенных базовых классов.

Опыт программирования показывает, что любой методический подход в технологии программирования не должен применяться слепо с игнорированием других подходов. Это относится и к объектно-ориентированному подходу. Существует ряд типовых проблем, для которых его полезность наиболее очевидна, к таким проблемам относятся, в частности, задачи имитационного моделирования, программирование диалогов с пользователем. Существуют и задачи, в которых применение объектного подхода ни к чему, кроме излишних затрат труда, не приведет. В связи с этим наибольшее распространение получили объектно-ориентированные языки программирования, позволяющие сочетать объектный подход с другими методологиями. В некоторых языках и системах программирования применение объектного подхода ограничивается средствами интерфейса с пользователем (например, Visual FoxPro ранних версий).

Наиболее используемыми в настоящее время объектно-ориентированными языками являются Паскаль с объектами и С++, причем наиболее развитые средства для работы с объектами содержатся в С++.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

**2 ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ**

В данной части будут рассмотрены 3 языка : C++, Java, Delphi.

**2.1 C++**

**C++** — компилируемый статически типизированный язык программирования общего назначения. Поддерживая разные парадигмы программирования, сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком Си, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования. Название «C++» происходит от языка C (Си), в котором унарный оператор ++ обозначает инкремент переменной.

Являясь одним из самых популярных языков программирования, C++ широко используется для разработки программного обеспечения. Область его применения включает операционные системы, прикладные программы, драйверы устройств, приложения для встраиваемых систем, высокопроизводительные серверы, а также развлекательные приложения, например, видеоигры. Существует несколько реализаций языка, как бесплатных, так и коммерческих. Их производят GNU Project, Microsoft, Intel иEmbarcadero (Borland). C++ оказал огромное влияние на другие языки программирования, в первую очередь на C# и Java.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

При создании C++ стремились сохранить совместимость с языком C. Большинство программ на C будут исправно работать и с компилятором C++. C++ имеет синтаксис, основанный на синтаксисе C.

В книге «Дизайн и эволюция C++» Бьёрн Страуструп описывает принципы, которых он придерживался при проектировании C++.Эти принципы объясняют, почему C++ именно такой, какой он есть. Некоторые из них:

* Получить универсальный язык со статическими типами данных, эффективностью и переносимостью языка Си.
* Непосредственно и всесторонне поддерживать множество стилей программирования, в том числе процедурное программирование, абстракцию данных, объектно-ориентированное программирование и обобщённое программирование.
* Дать программисту свободу выбора, даже если это даст ему возможность выбирать неправильно.
* Максимально сохранить совместимость с Си, тем самым делая возможным лёгкий переход от программирования на Си.
* Избежать разночтений между Си и C++: любая конструкция, которая допустима в обоих этих языках, должна в каждом из них обозначать одно и то же и приводить к одному и тому же поведению программы.
* Избегать особенностей, которые зависят от платформы или не являются универсальными.
* Никакое языковое средство не должно приводить к снижению производительности программ, не использующих его.
* Не требовать слишком усложнённой среды программирования.

**2.2 Java**

**Java** — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems. Приложения Java обычно компилируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине (JVM) независимо от компьютерной архитектуры. Дата официального выпуска — 23 мая 1995 года.

Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор, но с тем отличием, что байтовый код, в отличие от текста, обрабатывается значительно быстрее.

Достоинство подобного способа выполнения программ — в полной независимости байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят то, что исполнение байт-кода виртуальной машиной может снижать производительность программ и алгоритмов, реализованных на языке Java. Данное утверждение было справедливо для первых версий виртуальной машины Java, однако в последнее время оно практически потеряло актуальность. Этому способствовал ряд усовершенствований:

* применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде,
* широкое использование платформенно-ориентированного кода (native-код) в стандартных библиотеках,
* аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами фирмы ARM).

По данным сайта shootout.alioth.debian.org, для семи разных задач время выполнения на Java составляет в среднем в полтора-два раза больше, чем для C/C++, в некоторых случаях Java быстрее, а в отдельных случаях в 7 раз медленнее. С другой стороны, для большинства из них потребление памяти Java-машиной было в 10-30 раз больше, чем программой на C/C++.

Идеи, заложенные в концепцию и различные реализации среды виртуальной машины Java, вдохновили множество энтузиастов на расширение перечня языков, которые могли бы быть использованы для создания программ, исполняемых на виртуальной машине. Эти идеи нашли также выражение в спецификации общеязыковой инфраструктуры CLI, заложенной в основу платформы.NET компанией Microsoft.

Внутри Java существуют несколько основных семейств технологий:

* J2EE или Java EE (начиная с v1.5) — Java Enterprise Edition, для создания программного обеспечения уровня предприятия;
* J2SE или Java SE (начиная с v1.5) — Java Standard Edition, для создания пользовательских приложений, в первую очередь — для настольных систем;

Изм.

Лист

№ докум.

Подпись

Дата

Лист

* J2ME, Java ME или Java Micro Edition, для использования в устройствах, ограниченных по вычислительной мощности, в том числе мобильных телефонах, КПК, встроенных системах;
* Java Card для использования в устройствах без собственного человеко-машинного интерфейса, например в смарт-картах.

Самыми популярными считаются серверные технологии семейства J2EE.

Последним релизом является версия 1.6, в которой было произведено улучшение системы безопасности, улучшение поддержки XML, а также добавлена поддержка скриптового языка JavaScript на основе механизма Mozilla Rhino, улучшена интеграция с рабочим столом, добавлены некоторые новые возможности в создании графических интерфейсов.

Компанией Microsoft была разработана собственная реализация JVM (MSJVM), включавшаяся в состав различных операционных систем, начиная с Windows 98 (также входила в Internet Explorer от версии 3 и выше, что позволяло использовать MSJVM в ОС Windows 95 и Windows NT 4 после установки IE3+ на данные ОС).

MSJVM имела существенные отличия от Sun Java, во многом ломающие основополагающую концепцию переносимости программ между разными платформами:

* отсутствие поддержки программного интерфейса вызова удаленных методов (RMI);
* отсутствие поддержки технологии JNI;
* наличие нестандартных расширений, таких как средства интеграции Java и DCOM, работающих только на платформе Windows;
* JIT-компилятор.

Тесная интеграция Java с DCOM и Win32 ломала многоплатформенную суть языка и являлась усилием Microsoft по поглощению технологии (см. Критика Microsoft). Впоследствии это явилось поводом для судебных исков со стороны Sun Microsystems к Microsoft. Суд принял сторону компании Sun Microsystems. В конечном счёте между двумя компаниями была достигнута договорённость о возможности продления срока официальной поддержки пользователей нестандартной Microsoft JVM до конца 2007 года, но не более.

В 2005 году компанией Microsoft для платформы .NET был представлен Java-подобный язык J#, не соответствующий официальной спецификации языка Java и исключённый впоследствии из стандартного инструментария разработчика Microsoft Visual Studio, начиная с Visual Studio 2008.

**2.3 Delphi**

**Delphi** — язык программирования, который используется в одноимённой среде разработки. Сначала язык назывался Object Pascal. Начиная со среды разработки Delphi 7.0, в официальных документах Borland стала использовать название Delphi для обозначения языка Object Pascal.

Изначально среда разработки была предназначена исключительно для разработки приложений Microsoft Windows, затем был реализован также для платформ Linux(как Kylix), однако после выпуска в 2002 году Kylix 3 его разработка была прекращена, и, вскоре после этого, было объявлено о поддержке Microsoft .NET. При этом высказывались предположения, что эти два факта взаимосвязаны.

Реализация среды разработки проектом Lazarus (Free Pascal, компиляция в режиме совместимости с Delphi) позволяет использовать его для создания приложений на Delphi для таких платформ, как Linux, Mac OS X и Windows CE.

Также предпринимались попытки использования языка в проектах GNU и написания компилятора для GCC.http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8\_%D0%BD%D0%B0\_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8

Delphi — результат развития языка Турбо Паскаль, который, в свою очередь, развился из языка Паскаль. Паскаль был полностью процедурным языком, Турбо Паскаль, начиная с версии 5.5, добавил в Паскаль объектно-ориентированные свойства, а в Object Pascal — динамическую идентификацию типа данных с возможностью доступа к метаданным классов (то есть к описанию классов и их членов) в компилируемом коде, также называемом интроспекцией — данная технология получила обозначение RTTI. Так как все классы наследуют функции базового класса TObject, то любой указатель на объект можно преобразовать к нему, после чего воспользоваться методом ClassType и функцией TypeInfo, которые и обеспечат интроспекцию.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

Также отличительным свойством Object Pascal от С++ является то, что объекты по умолчанию располагаются в динамической памяти. Однако можно переопределить виртуальные методы NewInstance и FreeInstance класса TObject. Таким образом, абсолютно любой класс может осуществить «желание» «где хочу — там и буду лежать». Соответственно организуется и «многокучность».

Object Pascal (Delphi) являются результатом функционального расширения Turbo Pascal.

Delphi оказал огромное влияние на создание концепции языка C# для платформы .NET. Многие его элементы и концептуальные решения вошли в состав С#. Одной из причин называют переход Андерса Хейлсберга, одного из ведущих разработчиков Дельфи, из компании Borland Ltd. в Microsoft Corp.

* Версия 1 была предназначена для разработки под 16-разрядную платформу Win16;
* Версии со второй компилируют программы под 32-разрядную платформу Win32;
* Вместе с 6-й версией Delphi вышла совместимая с ним по языку и библиотекам среда Kylix, предназначенная для компиляции программ под операционную систему Linux;
* Версия 8 способна генерировать байт-код исключительно для платформы .NET. Это первая среда, ориентированная на разработку мультиязычных приложений (лишь для платформы .NET);
* Последующие версии (обозначаемые годами выхода, а не порядковыми номерами, как это было ранее) могут создавать как приложения Win32, так и байт-код для платформы .NET.

Delphi for .NET — среда разработки Delphi, а также язык Delphi (Object Pascal), ориентированные на разработку приложений для .NET.

Первая версия полноценной среды разработки Delphi для .NET — Delphi 8. Она позволяла писать приложения только для .NET.

В настоящее время, в Delphi 2006, можно писать приложения для .NET, используя стандартную библиотеку классов .NET, VCL для .NET. Среда также позволяет создавать .NET-приложения на C# и Win32-приложения на C++. Delphi 2006 содержит функции для написания обычных приложений с использованием библиотек VCL и CLX.

Delphi 2006 поддерживает технологию MDA с помощью ECO (Enterprise Core Objects) версии 3.0.

В марте 2006 года компания Borland приняла решение о прекращении дальнейшего совершенствования интегрированных сред разработки JBuilder, Delphi и C++ Builder по причине убыточности этого направления. Планировалась продажа IDE-сектора компании. Группа сторонников свободного программного обеспечения организовала сбор средств для покупки у Borland прав на среду разработки и компилятор.

Однако в ноябре того же года было принято решение отказаться от продажи IDE бизнеса. Тем не менее, разработкой IDE продуктов теперь будет заниматься новая компания — CodeGear, которая будет финансово полностью подконтрольна Borland.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

В августе 2006 года Borland выпустил облегченные версию RAD Studio под именем Turbo: Turbo Delphi (для Win32 и .NET), Turbo C#, Turbo C++.

В марте 2008 года было объявлено о прекращении развития этой линейки продуктов.

В марте 2007 года CodeGear порадовала пользователей обновленной линейкой продуктов Delphi 2007 for Win32 и выходом совершенно нового продукта Delphi 2007 for PHP.

В июне 2007 года CodeGear представила свои планы на будущее, то есть опубликовала так называемый roadmap.

25 августа 2008 года компания Embarcadero, новый хозяин CodeGear, опубликовала пресс-релиз на Delphi for Win32 2009.[7] Версия принесла множество нововведений в язык, как то:

* По умолчанию полная поддержка Юникода во всех частях языка, VCL и RTL; замена обращений ко всем функциям Windows API на юникодные аналоги (то есть MessageBox вызывает MessageBoxW, а не MessageBoxA).
* Обобщённые типы, они же generics.
* Анонимные методы.
* Новая директива компилятора $POINTERMATH [ON|OFF].
* Функция Exit теперь может принимать параметры в соответствии с типом функции.

В языке Object Pascal классы — это специальные типы данных, используемые для описания объектов. Соответственно объект, имеющий тип какого-либо класса, является экземпляром (instance) этого класса или переменной этого типа.

Класс представляет собой особый тип записи, имеющий в своем составе такие элементы (члены, member), как поля, свойства и методы. Поля класса аналогичны полям записи и служат для хранения информации об объекте. *Методами* называются процедуры и функции, предназначенные для обработки полей. *Свойства* занимают промежуточное положение между полями и методами.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

**ЗАКЛЮЧЕНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

Ведущие продавцы ПО для персоналок являются активными пользователями объектно-ориентированных языков и средств. Microsoft считается создателем объектно-ориентированных иерархий классов для Word и Excel.

Позиция Microsoft стала ясна после нескольких публикаций и интервью. Компания поддержала объектно-ориентированные расширения существующих языков, включая Basic. Bill Gates также поразил всех заявлением о том, что он не видит будущего за введением новых объектно-ориентированных языков после 1988 г., но ОС будущего должны иметь объектно-ориентированную поддержку: "Вы не поднимите всю глубину прикладных задач, работая только с ними, до тех пор, пока мы не сделаем это внутри ОС" [Byte, 1988]. Это будет достигнуто дополнительными возможностями OS/2.

Практически все объектно-ориентированные языки программирования являются развивающимися языками, их стандарты регулярно уточняются и расширяются. Следствием этого развития являются неизбежные различия во входных языках компиляторов различных систем программирования. Наиболее распространенными в настоящее время являются системы программирования Microsoft C++ , Microsoft Visual C++ и системы программирования фирмы Borland International.

Список литературы

**СПИСОК ЛИТЕРАТУРЫ**

1. А. Г. Иванов «Объектно-ориентированные системы: Состояние и перспективы», Москва 1992 г.
2. Подбельский В.В. Язык С++. – М.:Финансы и статистика, 1996.
3. http://ru.wikipedia.org

Изм.

Лист

№ докум.

Подпись

Дата

Лист