Содержание:

Введение

На современном этапе развития компьютерных технологий невозможно представить какого–либо высококвалифицированного специалиста, не владеющего информационными технологиями. Поскольку деятельность любого специалиста в значительной степени зависит от уровня владения информации, а также способности эффективно ее использовать. Для свободной ориентации в информационных потоках субъект любого профиля должен уметь получать, обрабатывать и использовать информацию, прежде всего, с помощью компьютеров, а также телекоммуникаций и других новейших средств связи.

Бурное развитие информационных технологий повлекло за собой создание множества искусственных языков, ориентированных на решение проблемы общения человека с компьютером. Это обусловлено тем, что прогресс компьютерных технологий определил процесс появления новых разнообразных знаковых систем для записи алгоритмов – языков программирования.

Язык программирования (ЯП) – формальная знаковая система, предназначенная для записи компьютерных программ, которая определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под ее управлением.

Так как, спектр программ высокого уровня многообразен и велик, в данной курсовой работе будут рассмотрены и проанализированы лишь те языки программирования которые, внесли значительный вклад в развитие IT.

1. Языки программирования (ЯП)

Программирование – это искусство создавать программные продукты, которые написаны на языке программирования.

Язык программирования (англ. Programming language) – система обозначений для описания алгоритмов и структур данных, определенная искусственная формальная система, средствами которой можно выражать алгоритмы. Язык программирования

определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполняет исполнитель (компьютер) под ее управлением.

Со времени создания первых программируемых машин было создано более двух с половиной тысяч языков программирования. Ежегодно их число пополняется новыми. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты обычно применяют в своей работе несколько языков программирования.

Задача данной работы – это рассмотрение языков программирования высокого уровня, поэтому, минуя языки низкого уровня, перейдем к обзору языков высокого уровня.

Если вкратце, то про языки высокого уровня можно сказать, что они более понятны человеку, чем компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому созданные программы легко переносятся с компьютера на компьютер. В основном достаточно просто перекомпилировать программу под определенную компьютерную архитектурную и операционную систему. Разрабатывать программы на таких языках гораздо проще и ошибок допускается меньше. Значительно сокращается время разработки программы, что особенно важно при работе над большими программными проектами.

К языкам программирования высокого уровня относятся:

- C
- C++
- C#
- Pascal
- Delphi
- Java
- Java Script
- Python

Недостатком языков высокого уровня является больший размер программ по сравнению с программами на языке низкого уровня. Поэтому в основном языки высокого уровня используются для разработок программного обеспечения компьютеров и устройств, которые имеют большой объем памяти. А разные подвиды ассемблера применяются для программирования других устройств, где

критичным является размер программы.

2. Язык программирования «C»

2.1 История создания

Язык **«C»** был создан в начале 70х годов, когда Кен Томпсон и Дэннис Ритчи из «Bell Labs» разрабатывали операционную систему UNDC. Сначала они создали часть компилятора **«C»**, затем использовали ее для компиляции остальной части компилятора **«C»** и, наконец, применили полученный в результате компилятор для компиляции UNIX. Операционная система UNIX первоначально распространялась в исходных кодах на **«C»** среди университетов и лабораторий, а получатель мог откомпилировать исходный код на **«C»** в машинный код с помощью подходящего компилятора **«C»**.

Распространение исходного кода сделало операционную систему UNIX уникальной; программист мог изменить операционную систему, а исходный код мог быть перенесен с одной аппаратной платформы на другую. Сегодня стандарт POSIX определяет стандартный набор системных вызовов UNIX, доступных в **«C»**, которые должны быть реализованы в версиях UNIX, являющихся POSIX-совместимыми.

Широкое распространение **языка «C»** на различных типах компьютеров (иногда называемых аппаратными платформами) привело, к сожалению, ко многим вариациям языка. Они были похожи, но несовместимы друг с другом. Это было серьезной проблемой для разработчиков программ, нуждавшихся в написании совместимых программ, которые можно было бы выполнять на нескольких платформах.

В 1983г. ANSI (Американский Национальный Комитет Стандартов) сформировал технический комитет X3J11 для создания стандарта языка **«C»** (чтобы "обеспечить недвусмысленное и машинно-независимое определение языка"). В 1989 стандарт был утвержден. ANSI скооперировался с ISO (Международной Организацией Стандартов), чтобы стандартизовать **«C»** в международном масштабе.

Совместный стандарт был опубликован в 1990 году и назван ANSI/ISO 9899:1990. Этот стандарт совершенствуется до сих пор и поддерживается большинством фирм разработчиков компиляторов.

2.2 Основные особенности языка

По сравнению с более ранним языком – BCPL, **«C»** был улучшен путем добавления типов данных определенной длины. Например, тип данных **int** может применяться для создания переменной с определенным числом битов (обычно 16), в то время как тип данных **long** может использоваться для создания целой переменной с большим числом битов (обычно 32). В отличие от других языков высокого уровня, **«C»** может работать с адресами памяти напрямую с помощью указателей и ссылок.

Поскольку **«C»** сохранил способность прямого доступа к аппаратному обеспечению, его часто относят к языкам среднего уровня или в шутку называют "мобильным языком ассемблера".

Что касается грамматики и синтаксиса, то **«C»** является структурным языком программирования. В то время как многие современные программисты мыслят в категориях классов и объектов, программисты на **«C»** думают в категориях процедур и функций. В **«C»** можно определить собственные абстрактные типы данных, используя ключевое слово **struct**. Аналогично можно описывать собственные целые типы (перечисления) и давать другие названия существующим типам данных при помощи ключевого слова **typedef**. В этом смысле **«C»** является структурным языком с зародышами объектно-ориентированного программирования.

Язык программирования **«C»** отличается минимализмом. Авторы языка хотели, чтобы программы на нём легко компилировались с помощью однопроходного компилятора, чтобы каждой элементарной составляющей программы после компиляции соответствовало весьма небольшое число машинных команд, а использование базовых элементов языка не задействовало библиотеку времени выполнения. Однопроходный компилятор компилирует программу, не возвращаясь назад к уже обработанному тексту. Поэтому использованию функций и переменных должно предшествовать их объявление. Код на **«C»** можно легко писать на низком уровне абстракции, почти как на ассемблере. **«C»** часто называют языком среднего уровня или даже низкого уровня, учитывая то, как близко он работает к реальным устройствам. Однако, в строгой классификации, он является языком высокого уровня.

«C» создавался с одной важной целью: сделать более простым написание больших программ с минимумом ошибок по правилам процедурного программирования, не

добавляя на итоговый код программ лишних накладных расходов для компилятора, как это всегда делают языки очень высокого уровня. С этой стороны **«C»** предлагает следующие важные особенности:

- простую языковую базу, из которой вынесены в библиотеки многие существенные возможности, вроде математических функций или функций управления файлами;
- ориентацию на процедурное программирование, обеспечивающую удобство применения структурного стиля программирования;
- систему типов, предохраняющую от бессмысленных операций;
- использование препроцессора для, например, определения макросов и включения файлов с исходным кодом;
- непосредственный доступ к памяти компьютера через использование указателей;
- минимальное число ключевых слов;
- передачу параметров в функцию по значению, а не по ссылке (при этом передача по ссылке эмулируется с помощью указателей);
- указатели на функции и статические переменные;
- области действия имён;
- структуры и объединения – определяемые пользователем собирательные типы данных, которыми можно манипулировать как одним целым;

В то же время в **«C»** отсутствуют:

- вложенные функции;
- сопрограммы;
- средства автоматического управления памятью;
- средства объектно-ориентированного программирования;
- средства функционального программирования.

После появления язык **«C»** был хорошо принят, потому что он позволял быстро создавать компиляторы для новых платформ, а также позволял программистам довольно точно представлять, как выполняются их программы. Благодаря этому программы, написанные на **«C»**, эффективнее написанных на многих других языках.

Одним из последствий высокой эффективности и переносимости **«C»** стало то, что многие компиляторы, интерпретаторы и библиотеки других языков высокого уровня часто написаны на языке **«C»**.

2.3 Область применения

Первоначально «**C**» задумывался как заменитель Ассемблера для написания операционных систем. Текст операционной системы оказывался легко переносимым с одной платформы на другую. Первой операционной системой, написанной практически целиком на **«C»**, была система Unix. В настоящее время почти все используемые операционные системы написаны на **«C»**. Тем не менее, область применения языка **«C»** не ограничилась разработкой операционных систем. **«C»** имеется множество систем программирования, позволяющих создавать программы, работающие в среде DOS, Windows и др. Он оказался очень удобен в программах обработки текстов и изображений, в научных и инженерных расчетах.

3. Язык программирования «C++»

3.1 История создания

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

Язык программирования **«C++»** был создан в начале 1980-х годов, его создателем стал сотрудник фирмы Bell Laboratories – Бьёрн Страуструп.

Он придумал ряд усовершенствований к языку программирования **«C»**, для собственных нужд. Т. е. изначально не планировалось создания языка программирования **«C++»**. Ранние версии языка **«C++»**, известные под именем **«Си с классами»**, начали появляться с 1980 года. Страуструп добавил возможность работы с классами и объектами, тем самым зародил предпосылки нового, основанного на синтаксисе **«C»**, языка программирования. Синтаксис **«C++»** был основан на синтаксисе **«C»**, так как Бьёрн Страуструп стремился

сохранить совместимость с языком **«C»**.

В 1983 году произошло переименование языка из «Си с классами» в «язык программирования C++».

3.2 Основные особенности языка

Язык программирования **«C++»** добавляет к **«C»** объектно-ориентированные возможности. Он вводит классы, которые обеспечивают три самых важных свойства ООП: инкапсуляцию, наследование и полиморфизм.

Методы класса – это функции, которые смогут применяться к экземплярам класса. Грубо говоря, метод – это функция, объявленная внутри класса и предназначенная для работы с его объектами. Методы объявляются в теле класса. Описываться могут там же, но могут и за пределами класса (внутри класса в таком случае достаточно представить прототип метода, а за пределами класса определять метод поставив перед его именем – имя класса и оператор.

Методы и поля входящие в состав класса называются членами класса. При этом методы часто называют функциями-членами класса.

Наследование:

В **«C++»** при наследовании одного класса от другого наследуется реализация класса, плюс класс-наследник может добавлять свои поля и функции или переопределять функции базового класса. Множественное наследование разрешено. Конструктор наследника вызывает конструкторы базовых классов, а затем конструкторы нестатических членов-данных, являющихся экземплярами классов. Деструктор работает в обратном порядке. Наследование бывает публичным, защищённым и закрытым.

Полиморфизм:

Целью полиморфизма, применительно к объектно-ориентированному программированию, является использование одного имени для задания общих для класса действий. Выполнение каждого конкретного действия будет определяться типом данных. Преимуществом полиморфизма является то, что он помогает снижать сложность программ, разрешая использование того же интерфейса для задания единого класса действий. Выбор же конкретного действия, в зависимости

от ситуации, возлагается на компилятор. Полиморфизм может применяться также и к операторам.

Инкапсуляция:

Основным способом организации информации в **«C++»** являются классы. В отличие от структуры (struct) языка **«C»**, которая может состоять только из полей и вложенных типов, класс (class) **«C++»** может состоять из полей, вложенных типов и функций-членов. Инкапсуляция в **«C++»** реализуется через указание уровня доступа к членам класса: они бывают публичными (public), защищёнными (protected) и закрытыми (private). В **«C++»** структуры отличаются от классов тем, что по умолчанию члены и базовые классы у структуры публичные, а у класса — собственные.

В отличии от языка **«C»** в **«C++»** добавились новые возможности:

- поддержка объектно-ориентированного программирования;
- поддержка обобщённого программирования через шаблоны;
- дополнительные типы данных;
- исключения;
- пространства имён;
- встраиваемые (inline) функции;
- перегрузка операторов;
- перегрузка функций;
- ссылки и операторы управления свободно распределяемой памятью;
- дополнения к стандартной библиотеке.

В C++ не разрешается:

- вызывать функцию main() внутри программы, в то время как в **«C»** это действие правомерно;
- неявное приведение типов между несвязанными типами указателей;
- использовать функции, которые ещё не объявлены.

3.3 Область применения

Язык программирования **«C++»** широко используется для разработки программного обеспечения. А именно, создание разнообразных прикладных программ, разработка операционных систем, драйверов устройств, а также видео

игр и многое другое. Существует несколько реализаций языка программирования **«C++»** – как бесплатных, так и коммерческих. Их производят проекты: GNU, Microsoft и Embarcadero (Borland). Проект GNU – проект разработки свободного программного обеспечения (СПО).

4. Язык программирования «C#»

4.1 История создания

К 2000 году компания Microsoft подготовила промышленные версии новых компонентных технологий и решений в области обмена сообщениями и данными, а также создания Internet-приложений (COM+, ASP+, ADO+, SOAP, Biztalk Framework). В поддержку этих новшеств Microsoft выпустила инструментарий для разработки приложений – платформу .NET. Она также объединяла «под одной крышей» несколько языков программирования, что было в новинку для того времени.

Еще одним новшеством платформы .NET была технология активных серверных страниц ASP.NET (Active Server Page). С её помощью можно было относительно быстро разработать веб-приложения, взаимодействующие с базами данных.

Язык программирования **«C#»** был создан специально для ASP.NET. На **«C#»** полностью была написана, и сама ASP.NET.

Название «Си шарп» (от англ. sharp – диез) несет «сакральный» смысл. Знак «#» (в музыкальной нотации читается как «диез») означает повышение высоты звука на полтона. С другой стороны, название **«C#»** получается путем следующей «эволюционной цепочки»: C → C++ → C++++(C#), так как символ «#» можно составить из 4-х знаков «+».

Вследствие технических ограничений на отображение (стандартные шрифты, браузеры и т. д.) и того, что знак диез **«#»** не представлен на стандартной клавиатуре, знак **«#»** был выбран для представления знака диез при записи имени языка программирования. Это соглашение отражено в Спецификации Языка **«C#»** ECMA-334. Названия языков программирования не принято переводить, поэтому язык следует называть по-английски «Си шарп».

Авторами этого языка программирования стали Скотт Вилтамут и Андерс Хейльсберг – создатель Турбо Паскаля и Дельфи, перешедший в 1996 году в Microsoft.

По одной из версий, он вынашивал замысел нового языка и даже новой платформы (которая сейчас носит название .NET), еще работая в компании Borland.

«C#» поддерживает все три «столпа» объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. Кроме того, в нем была реализована автоматическая «сборка мусора», обработки исключений, динамическое связывание.

4.2 Основные особенности языка

Язык **«C#»** простой, современный объектно-ориентированный и типобезопасный язык программирования. **«C#»** относится к широкоизвестному семейству языков **«C»**, и покажется хорошо знакомым любому, кто работал с «C», «C++», «Java» или «JavaScript».

«C#» является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Важная особенность таких компонентов – это модель программирования на основе свойств, методов и событий. Каждый компонент имеет атрибуты, предоставляющие декларативные сведения о компоненте, а также встроенные элементы документации. **«C#»** предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому **«C#»** отлично подходит для создания и применения программных компонентов.

Вот лишь несколько функций языка **«C#»**, обеспечивающих надежность и устойчивость приложений:

- сборка мусора автоматически освобождает память, занятую уничтоженными и неиспользуемыми объектами;
- обработка исключений, что дает структурированный и расширяемый способ выявлять и обрабатывать ошибки;

- строгая типизация языка не позволяет обращаться к неинициализированным переменным, выходить за пределы массива или выполнять неконтролируемое приведение типов.

В C# существует единая система типов. Все типы **«C#»**, включая типы-примитивы, такие как **int** и **double**, наследуют от одного корневого типа **object**. Таким образом, все типы используют общий набор операций, и значения любого типа можно хранить, передавать и обрабатывать схожим образом.

Кроме того, **«C#»** поддерживает пользовательские ссылочные типы и типы значений, позволяя как динамически выделять память для объектов, так и хранить упрощенные структуры в стеке.

Чтобы обеспечить совместимость программ и библиотек **«C#»** при дальнейшем развитии, при разработке **«C#»** много внимания было уделено управлению версиями. Многие языки программирования обходят вниманием этот вопрос, и в результате программы на этих языках ломаются чаще, чем хотелось бы, при выходе новых версий зависимых библиотек. Вопросы управления версиями существенно повлияли на такие аспекты разработки **«C#»**, как раздельные модификаторы **virtual** и **override**, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

4.3 Область применения

Язык **«C#»** удобен для написания интерфейсов, в том числе в **web**, а также для разработки различных сервисов и служб, где важна также и скорость разработки. **«C#»** позволяет быстрее разрабатывать бизнес-приложения. Однако приложения будут работать под управлением среды .Net Framework (Mono для Linux). Большинство типичных потребностей бизнеса вполне можно реализовать с помощью **«C#»**.

5. Язык программирования «Pascal»

5.1 История создания

Язык Паскаль был разработан в 1970 г. Никлаусом Виртом как язык, обеспечивающий строгую типизацию и интуитивно понятный синтаксис. Он был

назван в честь французского математика, физика и философа Блеза Паскаля.

Одной из целей создания языка Паскаль Никлаус Вирт считал обучение студентов структурному программированию. До сих пор Паскаль заслуженно считается одним из лучших языков для начального обучения программированию. Его современные модификации, такие как Object Pascal, широко используются в промышленном программировании (среда Delphi).

Наиболее популярным решением для персональных компьютеров в 80-е - начале 90 годов стал компилятор и интегрированная среда разработки Turbo Pascal фирмы Borland. Встроенный компилятор обеспечивал высокую скорость компиляции и высокое качество кода (отсюда приставка Turbo). Среда Turbo Pascal обеспечивала также отладку кода, содержала богатый набор примеров. Все эти качества позволили Turbo Pascal стать стандартом Паскаля де-факто.

Выпущенная в 1995 г. как продолжение среды Turbo Pascal система программирования Delphi стала одной из лучших сред для быстрого создания приложений. Delphi ввела в язык Паскаль ряд удачных объектно-ориентированных расширений; обновленный язык получил название Object Pascal. Начиная с версии Delphi 7.0, язык Delphi Object Pascal стал называться просто Delphi, однако, старое название используется часто. Последняя версия среды - Delphi XE.

Наиболее известной свободной реализаций языка Паскаль является Free Pascal. Помимо открытости исходного кода, его основным преимуществом является мультиплатформенность, а также поддержка различных диалектов Паскаля. На основе FreePascal создана свободная мультиплатформенная среда Lazarus, аналогичная среде Delphi. Однако, бедный и не меняющийся десятилетиями консольный интерфейс интегрированной среды Free Pascal, мало совместимый с современными интерфейсами рабочих столов операционных систем, всё более отталкивает обучаемых, неправильно формируя у них представление, что Паскаль - устаревший язык.

Одним из ярких событий, связанных с развитием языка Паскаль, стало появление языка и компилятора Oxygene фирмы RemObjects, который создатели заслуженно назвали современным Паскалем 21 века. Oxygene может генерировать код под различные платформы, в том числе под платформы .NET и Java. Основным его недостатком является отсутствие бесплатного компилятора и среды для образовательных целей. Кроме того, Oxygene достаточно сильно отличается от канонического языка Паскаль (методы классов вместо процедур и функций), что

отражает его сугубо профессиональную направленность.

Язык и система программирования PascalABC.NET призваны изменить сложившуюся ситуацию и вернуть языку Паскаль былую привлекательность как для обучения, так и для профессионального программирования, помножив ее на мощь платформы .NET.

5.2 Основные особенности языка

Pascal является традиционным алгоритмическим языком программирования, продолжающим линию Algol-60. Это означает, что программа на языке Pascal представляет собой специально организованную последовательность шагов по преобразованию данных, приводящую к решению некоторой задачи. Это отличает Pascal от так называемых непроцедурных языков типа Prolog, по существу, представляющих собой формализмы для записи начальных условий некоторой задачи и синтезирующих решение посредством встроенных механизмов логического вывода.

Язык Pascal содержит удобные средства для представления данных. Развитая система типов позволяет адекватно описывать данные, подлежащие обработке, и конструировать структуры данных произвольной сложности. Pascal является типизированным языком, что означает фиксацию типов переменных при их описании, а также строгий контроль преобразований типов и контроль доступа к данным в соответствии с их типом (как на этапе компиляции, так и при исполнении программ).

Набор операторов языка Pascal отражает принципы структурного программирования и позволяет записывать достаточно сложные алгоритмы в компактной и элегантной форме. Pascal является процедурным языком с традиционной блочной структурой и статически определенными областями действия имен. Процедурный механизм сочетает в себе простоту реализации и использования и гибкие средства параметризации.

Синтаксис языка достаточно несложен. Программы записываются в свободном формате, что позволяет сделать их наглядными и удобными для изучения.

Паскаль – компилятор, тоесть, прежде чем начать испоолнение программы, Паскаль полностью прочитывает исходный текст, написанный программистом, и составляет последовательность машинных кодов, выполняющую те действия, которые описал программист в hqundmnl тексте. Эта последовательность

сохраняется в файл с расширением “.EXE” и является самостоятельным исполняемым файлом, который может быть запущен сам по себе, уже без участия Паскаля и, даже, на другом компъютере, на котором Паскаль может быть не установлен.

5.3 Область применения

К сожалению, несмотря на свои особенности, Pascal в наши дни не добился такой популярности, как языки линейки «C».

Спецификой данного языка программирования являются:

- Сравнительно простой синтаксис;
- Мизерный объём базовых понятий;
- Программы, написанные на Pascal е легко читаемые;
- Довольно низкие системные и аппаратные требования компилятора и программ, написанных при помощи Pascal;
- Универсальность языка;
- Язык Pascal можно использовать для решения чуть ли не всех задач по программированию;
- Возможность программирования «сверху-вниз», объектно-ориентированного и структурного программирования.

Именно поэтому в наши дни Pascal используется во многих учебных заведениях в качестве изучения языка программирования, как основа для начинающих программистов.

6. Язык программирования «Delphi»

6.1 История создания

Язык Delphi берет свое начало от языка программирования Pascal, названного в честь знаменитого французского математика Блеза Паскаля . Язык Pascal был создан Никлаусом Виртом в Цюрихе как учебный язык компьютерного программирования. В результате Pascal быстро получил широкую популярность и стал основным учебным языком во многих университетах как в США, так и во всем мире.

Компанией Borland была разработана популярная версия этого языка- Turbo Pascal. По мере развития операционных систем Windows и распространения концепции объектно-ориентированного программирования язык Pascal был естественным образом расширен до Turbo Pascal for Windows и Object Pascal for Windows.

Следующим естественным шагом было создание Delphi- среды разработки программ на Object Pascal.В систему Delphi входят компилятор с Object Pascal, визуальная среда разработки, инструменты взаимодействия с базами данных и библиотека VCL.

Версия Delphi была выпущена в феврале 1995 года, Delphi 2- в марте 1996 года, а Delphi 3- в мае 1997 года. Затем было решено интегрировать эту платформу программирования с CORBA, быстро развивающейся технологией создания распределенных приложений. В середине 1997 года компанией Borland была приобретена компания Visigenis, известная своими разработками в области стандартной промышленной технологии ORB. Благодаря этому в июне 1998 года на рынко появилась версия Delphi 4, обладающая встроенными средствами поддержки технологии CORBA. Версия Delphi 5 была выпущена в августе 1999 года. Новая версия, Delphi 6, была выпущенная в мае 2001 года, за ней вышли Delphi 7, 2002 год и, наконец, Delphi 8- в начале 2004 года.

Существует четыре варианта постановки Delphi 7: это пакеты Personal , Professional , Enterprise и Architect . Они отличаются один от другого функциональными возможностями и предоставляемым набором компонентов и ресурсов.

Особо следует отметить, что Delphi 7 и Delphi 8 позволяют работать со средой .Net, а также создавать программы для среды Linux. Однако в последнем случае выполняемый код среды Linux не создается- для его получение потребуется дополнительно откомпилировать созданные в Delphi программы в среде Kylix.

6.2 Основные особенности языка

Процесс разработки в Delphi предельно упрощен. В первую очередь это относится к созданию интерфейса, на который уходит 80% времени разработки программы. Достаточно просто перетащить нужные компоненты на поверхность Windows-окна (в Delphi оно называется формой) и настраиваете их свойства с помощью специального инструмента (Object Inspector). С его помощью можно связать события этих компонентов (нажатие на кнопку, выбор мышью элемента в списке и т.д.) с кодом его обработки.

Разработчик может использовать мощные средства отладки (вплоть до пошагового выполнения команд процессора), удобную контекстную справочную систему (в том числе и по Microsoft API), средства коллективной работы над проектом.

В Delphi современных версий возможно создавать компоненты ActiveX без использования Microsoft IDL, расширять возможности web-сервера (скрипты на стороне сервера), практически ничего не зная об HTML, XML или ASP.

Есть возможность создавать распределенные приложения на базе COM и CORBA, Интернет- и intranet-приложения, используя для доступа к данным Borland DataBase Engine, ODBC-драйверы или Microsoft ADO.

В Delphi разработчикам дали возможность создавать свои собственные компоненты, импортировать OCX-компоненты, создавать шаблоны проектов и мастеров, генерирующих заготовки проектов. Более того, авторы предоставили разработчику интерфейс для связи других приложений (или внешних программ) с Delphi IDE.

6.3 Область применения

Изначально среда разработки была предназначена исключительно для разработки приложений Microsoft Windows, затем был реализован также для платформ GNU/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) позволяет использовать его для создания приложений на Delphi для таких платформ, как GNU/Linux, Mac OS X и Windows CE.

7. Язык программирования «JAVA»

7.1 История создания

Приблизительно в 1990 Джеймс Гослинг, Билл Джой, Патрик Ногтон и другие в Sun Microsystems начали разрабатывать язык по имени Oak. Прежде всего они видели применение Java для встроенных микрокомпьютерных модулей бытовой техники, в видеомагнитофонах, тостерах, а также для PDA (personal data assistants).

Чтобы решать эти задачи, Oak должен был быть:

- Независимым от платформы (с тех пор как продукцию стали вовлекать многие изготовители)
- Чрезвычайно надежным
- Компактным.

Однако, в 1993 рынки интерактивного телевидения и PDA пошли на убыль. Тогда бурно развивался internet и сети. Так что Sun сдвинула целевой рынок в сторону internet-приложений и заменила название проекта на Java.

В основе Java лежат языки C и C++. Его синтаксис получается из C, а ориентированные на объект особенности влияет C++.

В 1994 Sun's выпустила браузер HotJava. Он был написан на Java за несколько месяцев, что показало мощность апплетов, программ которые работают в пределах браузера, а также для того, чтобы ускорить процесс разработки программ.

Развивающийся наряду с огромным интересом к internet, Java быстро получил широкое распространение, и ожидал роста для того, чтобы стать доминирующим языком программирования для написания приложений для потребителей и браузера. Однако, ранние версии Java не обладали широтой и глубиной возможностей, необходимых для приложений клиента (то есть потребителя). Например, графика в Java 1.0 казалась грубой и неуклюжей по сравнению программным обеспечением, разработанным на C и других языках.

Наряду с тем, что Java отставал в развитии клиентских приложений, он стал очень популярным языком для развития предприятий, или микропрограммных средств, приложений типа интерактивной памяти, диалоговых обработок запросов, интерфейсов базы данных, и т.д. Java также стал весьма обычным на небольших платформах типа сотовых телефонов и PDAs.

7.2 Основные особенности языка

Java - полностью объектно-ориентированный язык программирования. В Java отсутствует понятие процедур. С помощью Java мы можем решить различные задачи и тот же самый круг проблем, что и на других языках программирования. Java может использоваться для создания двух типов программ: Приложений и Апплетов. Приложение - программа, которая выполняется на нашем компьютере, под его операционной системой. Приложения Java могут быть непосредственно

выполнены, используя интерпретатор Java. Апплет - небольшая программа работающая с окнами, которые внедрены в страницу HTML. Чтобы выполнить Java апплеты, нужна поддержка Java Web-браузером, то есть Internet Explorer, Netscape Navigator, Hot Java и т.д. или средство просмотра апплета. Также Java допускал другие средства, с помощью которых браузер мог выполнить программу Java на нашей системе.

Java - это интерпретируемый и компилированный язык программирования. Исходный текст (файлы с расширением a Java) откомпилирован со справкой компилятора Java (javac), который преобразовывает исходный текст в байт-код (файлы с расширением a.class). Цель проектировщиков Java состояла в том, чтобы разработать язык, посредством которого программист мог записать код, который мог бы выполняться всегда, в любое время.

Основными особенностями языка Java являются:

- Простой
- Объектно-ориентированный
- Распределённый
- Устойчивый
- Безопасный
- Независимый от структуры системы
- Мобильный
- Интерпретирующее выполнение
- Высокая эффективность
- Многопоточный
- Динамичный

Рассмотри их по подробнее.

Простота

Проектировщики Java пытались разработать язык, который могли бы быстро изучить программисты. Также они хотели, чтобы язык был знаком большинству программистов, для простоты перехода. Отсюда, в Java проектировщиками было удалено множество сложных особенностей, которые существовали в «C» и «C++». Особенности, такие как манипуляции указателя, перегрузка оператора и т.д. в Java не существуют.

Java не использует **goto** инструкцию, а также не использует файлы заголовка. Конструкции подобно **struct** и **union** были удалены из Java.

Объектно-ориентированность

В Java всё может быть объектом. Так основное внимание уделяется свойствам и методам, которые оперируют данными в нашем приложении и нет концентрации только на процедурах. Свойства и методы вместе описывают состояние и поведение объекта. В Java мы будем наталкиваться на термин метод очень часто, с ним мы будем должны познакомиться. Термин метод используется для функций.

Распределенность

Java может использоваться для разработки приложений, которые работают на различных платформах, операционных системах и графических интерфейсов пользователя. Java предназначен также для поддержки сетевых приложений. Таким образом Java широко используется как инструмент разработки в среде подобной Internet, где существуют различные платформы.

Устойчивость

Java - язык со строгим контролем типов, так что требуется явное объявление метода. Java проверяет код во время трансляции и во время интерпретации. Таким образом устраняются некоторые типы ошибок при программировании. Java не имеет указателей и соответственно арифметических операций над ними. Все данные массивов и строк проверяются во время выполнения, что исключает возможность выхода за границы дозволенного. Преобразование объектов с одного типа на другой также проверяется во время выполнения.

Автоматическая обработка исключений - В традиционных средах программирования, программист должен был вручную распределять память, и в конце программы имел явное количество свободной памяти. Возникали проблемы, когда программист забывал освобождать память. В Java, программист не должен беспокоиться о проблеме, связанной с освобождением памяти. Это делается автоматически, поскольку Java обеспечивает обработка исключений для объектов, которые не используются.

Безопасность

Вирусы - большая причина беспокойства в мире компьютеров. До Java, программисты должны были сначала просмотреть файл перед загрузкой и

выполнением. Даже после этого они не были уверены в надёжности файла. Также, существует много специальных программ, о которых мы должны знать. Эти программы могут находить уязвимые данные нашей системы.

Java обеспечивает управляемую среду, в которой выполнена программа. Java никогда не предполагает, что код может быть безопасно выполнен. И так как Java - больше чем язык программирования, он обеспечивает несколько уровней контроля защиты. Со справкой этих уровней, он гарантирует безопасную среду выполнения.

Первый уровень - это безопасность, обеспеченная языком Java. Свойства и методы описываются в классе, и к ним можно обратиться только через интерфейс, обеспеченный классом. Java не позволяет никаких операций с указателями, таким образом запрещает прямой доступ к памяти. Избегается переполнение массивов. Проблемы, связанные с безопасностью и мобильностью, скрыты.

На следующем уровне компилятор, прежде чем приступить к компиляции кода, проверяет безопасность кода и затем следует в соответствии с протоколами, установленными Java.

Третий уровень - это безопасность, обеспеченная интерпретатором. Прежде, чем байт-код будет фактически выполнен, он является полностью укрытым верификатором.

Четвертый уровень заботится о загрузке классов. Загрузчик класса гарантирует, что класс не нарушает ограничения доступа прежде, чем он загружен в систему.

Независимость от структуры системы

Нейтралитет достигается при смешении трансляции и интерпретации.

Программы Java оттранслированы в байт-код компилятором

• байт-код - это универсальный машинный код

Байт-код выполняется интерпретатором (Виртуальная Машина Java)

• интерпретатор должен выполнять байт-код для каждой аппаратной платформы

• байт-код выполняется на любой версии Виртуальной Машины Java

Мобильность

Независимость от платформы означает лёгкость переноса программы с одного компьютера на другой компьютер без каких-либо трудностей. Также Java - платформа, независима на обоих уровнях, то есть на первичном (исходном) и на вторичном уровне.

Java - это язык со строгим контролем типов, что означает, что мы должны объявлять тип для каждой переменной. И эти типы данных в Java одинаковы для всех платформ. Java имеет свои собственные библиотеки фундаментальных классов, которые облегчают запись кода для программиста, который может быть перемещен с одной машины на другую, без потребности перезаписи кода. Короче говоря, независимость от платформы на исходном уровне означает, что мы можем переместить наш исходный текст из одной системы в другую, компилируя код, и работая в системе.

Платформа, независимая на вторичном уровне означает, что откомпилированный двоичный файл может быть выполнен на различных платформах, не перетранслируя код, если они имеют Виртуальную Машину Java, которая функционирует как интерпретатор.

Интерпретирующее выполнение

Java - это интерпретируемый язык. Это означает, что каждая команда оттранслирована в машинный код во время выполнения, а не в течение трансляции, а также позволяет перезаписывать и изменять программу, во время её выполнения.

Трансляция Java и процедура выполнения включают следующее:

- Различные исходные файлы обрабатываются компилятором javac, для получения множество файлов класса. Эти файлы содержат байт-код, который не зависит от архитектуры и платформы исполняющей его.
- Файлы класса Java могут быть выполнены со справкой загрузчика (интерпретатора), утилиты по имени java, которая функционирует, чтобы транслировать универсальные байт-коды Java в машинные выполняемые коды. Никакой компоновщик при этом не требуется.

Из-за его интерпретирующей процедуры выполнения, Java имеет следующие преимущества:

- Файлы класса Java могут быть выполнены на любой платформе при условии, что данная платформа имеет надлежащую утилиту загрузчика java.
- Файлы класса Java делают более эффективное использование памяти, нежели отдельные (часто большие) выполняемые программы, потому что файлы класса могут быть связаны загрузчиком на основании управляемого запроса.

Высокая Эффективность

Java был разработан, чтобы хорошо работать на центральных процессорах с очень низким энергопотреблением. Байт-код Java был тщательно продуман так, чтобы его можно было сразу непосредственно транслировать в машинный код с высокой эффективностью, используя компилятор.

Многопоточность

Встроенная поддержка многопоточности снабжает программистов Java мощным инструментом для улучшения интерактивной работы графических приложений. Если наше приложение должно выполнять мультипликацию и музыку игры при прокрутке страницы и загрузку текстового файла с сервера, то многопоточность - это способ быстро реализовать поставленную задачу.

Динамичность

Java - динамичный адаптированный язык программирования. Программы Java несут много информации во время выполнения, для проверки правильности обращения к объектам во время выполнения. Это свойство позволяет динамически безопасно связать код.

7.3 Область применения

Существует множество областей применения Java, от сайтов электронной коммерции до Android приложений, от научных до финансовых приложений, таких как трейдинговые системы, от игр, типа Minecraft, до настольных программных средств, таких как Eclipse, Netbeans и IntelliJ, от open source фреймворков до J2ME приложений и т.д. Давайте детальнее рассмотрим каждое из них.

Android приложения.

Если хотите увидеть, где используется Java, не нужно далеко идти. Просто возьмите свой телефон на Android, абсолютно все приложения написаны на Java, с использованием Google и Android API, которые схожи с JDK. Пару лет назад Android

предоставил необходимые возможности, благодаря чему сегодня многие Java программисты – Android разработчики. Кстати, Android использует другую JVM и другой и другой способ компановки, но код всё ещё написан на Java.

Серверные приложения в сфере финансовых услуг.

Java очень обширно применяется в финансовой сфере. Многие мировые инвестиционные банки, типа Goldman Sachs, Citigroup, Barclays, Standard Charted и другие используют Java для написания фронт-энд и бэк-энд офисных электронных систем, систем регулирования и конфирмации, проектов обработки данных и некоторых других. Преимущественно Java используется при написании серверных приложений, в большинстве своём без какого-либо пользовательского интерфейса, которые получают данные с одного сервера, обрабатывают их и отправляют дальше. Java Swing был также популярен для создания «толстоклиентных» интерфейсоф, но сейчас C# быстро захватывает рынок в этой области, а Swing уже выдыхается.

Вэб-приложения.

Также Java широко используется в электронной коммерции и в области вэб-приложений. Огромное количество RESTful сервисов было создано с использованием Spring MVC, Struts 2.0 и похожих фреймворков. Даже простейшие приложения, основанные на Servlet, JSP и Struts, достаточно популярны в различных государственных проектах. Многие вэб-приложения государственных, оздоровительных, страховых, образовательных, оборонительных и некоторых других отделений написаны на Java.

Программные средства.

Многие полезные програмные средства и средства разработки написаны и разработаны на Java, например Eclipse, IntelliJ Idea и Netbeans IDE. Мне кажется это, к тому же, наиболее используемые приложения, написанные на Java. Было время, когда Swing был очень популярен при создании «толстых клиентов», преимущественно в финансовой сфере. Сегодня Java FX набирает всё большую популярность, но это всё ещё не замена Swing, к тому же C# практически полностью вытеснил Swing из финансовой области.

Трейдинговые приложения.

Сторонние трейдинговые приложения, которые также часть большой индустрии финансовых услуг, тоже используют Java. Популярные приложения, типа Murex, которые используются во многих банках, написаны на Java.

J2ME приложения.

Несмотря на то, что появление iOS и Android практически уничтожило J2ME рынок, в мире ещё огромное количество дешёвых телефонов от Nokia и Samsung, использующих J2ME. Было время, когда практически все игры и приложения, доступные на Android, были написаны с использованием MIDP и CLDC, которые являются частью платформы J2ME. J2ME всё ещё популярен в таких средствах, как Blu-ray, карточки и телевизионные приставки. Одна из причин такой популярности WhatsApp – он также доступен на J2ME.

Встраиваемые системы.

Обширна Java и в области встраиваемых систем. Можно увидеть на что способна платформа, вам нужно всего 130 KB для использования Java (на смарт-картах и сенсорах). Изначально Java разрабатывалась для встраиваемых систем. В действительности эта область была частью начальной кампании Java «пиши один раз, запускай где-угодно» и похоже, что она приносит свои плоды.

Большие данные.

Hadoop и другие технологии обработки больших данных так или иначе используют Java, например Hbase и Accumulo от Apache, или ElasticSearch. Хоть Java и не доминирует в этой области, поскольку существуют такие технологии, как MongoDB, которые написаны на C++. У Java есть потенциал получить большую долю этой растущей области, если Hadoop или ElasticSearch расширятся.

Высокочастотные трейдинговые пространства.

Java улучшила свои эксплуатационные показатели и с современными JIT-ами она способна предоставить производительность на уровне C++. По этой причине Java популярна и при написании высокопроизводительных систем, потому что хоть производительность проигрывает в сравнении с родным языком, но вы можете пожертвовать безопасностью, мобильностью и надёжностью ради большей скорости и требуется всего один неопытный C++ программист, чтобы сделать приложение медленным и ненадёжным.

Научные приложения.

В наши дни часто Java – выбор по-умолчанию для научных приложений, включая обработку естественного языка. Основная причина в том, что Java более безопасна, мобильна и надёжна и имеет лучшие инструменты параллелизации , чем C++ и другие языки.

В девяностые Java была достаточно популярна в интернете, благодаря апплетам, но спустя годы, апплеты утратили свою популярность, преимущественно из-за различных проблем безопасности. В наши дни настольная Java и апплеты практически мертвы. Java по-умолчанию любимец в индустрии программного обеспечения, и широко используется в финансовой сфере, инвестиционных банках и в области электронной коммерции. Каждый, изучающий Java, имеет яркое будущее. Java 8 только укрепила веру в то, что Java продолжит доминировать в области разработки ещё долгие годы.

8. Язык программирования «JAVA Script»

8.1 История создания

События, в результате которых появился JavaScript, разворачивались в течение шести месяцев, с мая по декабрь 1995 года.

Фирма Netscape Communications Corporation с самого начала принимала заметное участие в динамичном развитии всемирной паутины. В борьбе за первенство на этом поприще она выдвинулась благодаря созданию и бесплатному распространению (для использования в домашних условиях) броузера Netscape Navigator.

Затем, через короткое время, Netscape создала скриптовый язык под названием LiveScript, призванный испонять роль чудесного средства, позволяющего превратить статичные документы в более-менее интерактивные. LiveScript, поддерживаемый первыми версиями броузера Netscape Navigator, пользовался большой популярностью и успехом. В то же время инженеры фирмы Sun Microsystems, которым надоело приспосабливать свое программное обеспечение к стандартам различных интерфейсов, разработали язык Java.

В результате соглашения между Netscape Communications и Sun Microsystems и объединения идей LiveScript со структурой Java появилась среда под названием

«Mocha», предназначенная для разработки сетевых приложений и, в конце концов, для создания динамичных web-страниц. Существенно, что она имела открытый характер и была независима от используемой программной платформы.

Проект завершился созданием спецификаций, которые были опубликованы двумя предприятиями в декабре 1995 года под названием JavaScript в версии 1.0. Новой технологией заинтересовались многие фирмы, которые хотели использовать язык JavaScript в своих продуктах. Фирма Microsoft дфже объявила, что язык JavaScript может обслуживаться броузером Internet Explorer.

Но у фирмы Microsoft были определенные трудности с использованием JavaScript, поскольку первые ее реализации этого языка, названного Jscript, в броузере Internet Explorer 3.0 были недостаточно надежны. Это вынудило Microsoft использовать распространенную, стандартную версию JavaScript, а Internet Explorer стал предоставлять возможность запуска скриптов на языке VBScript (Visual Basic Script), который является авторским решением Microsoft. Его синтаксис и возможности очень похожи на JavaScript. Однако, более универсальной и повсеместно используемой является технология JavaScript.

8.2 Основные особенности языка

Возможности JavaScript практически не ограниченны. Его встраивают в приложения, веб-страницы, сервисы и standalone-продукты. Новомодная и мощная связка AJAX привнесла ещё больше возможностей для реализации потенциала JavaScript. Именно она даёт возможность незаметно для пользователя обновлять небольшую часть страницы, не перезагружая её целиком. Это позволяет сэкономить трафик и увеличить удобство использования веб-интерфейсов.

Но как и у любой системы у JavaScript есть ряд недостатков связанных с безопасностью - использование так называемой атаки типа XSS. В её основе лежит внедрение скрипта в тело страницы, которая отображается пользователю. Код может получить права текущего посетителя и использовать их во вред, например, похитить личные данные.

Куки - небольшая часть данных, которая отправляется веб-приложением на компьютер пользователя. Применяется для идентификации посетителей, записи их предпочтений или просто для хранения настроек сайта. Это могут использовать и злоумышленники с помощью JavaScript. Получив доступ к кукис, хакер может авторизоваться на сайте под профилем пользователя и похитить личные данные.

Клиентское приложение, написанное на JavaScript, может подвергнуться обратной разработке, неправильно проходить авторизацию. Поэтому JavaScript не предоставляет средств по обеспечению достойного уровня безопасности.

8.3 Область применения

Область применения этого языка удивительно обширна и ничем не ограничена: среди программ, которые используют JS, присутствуют и тестовые редакторы, и приложения (как для компьютеров, так и мобильные и даже серверные), и прикладное ПО. С помощью него можно:

- Изменять страницы браузеров;
- Добавление или удаление тегов;
- Изменение стилей страницы;
- Информация о действиях пользователя на странице;
- Запрос доступа к случайной части исходного кода страницы;
- Внесение изменений в этот код;
- Выполнение действия с cookie-файлами.

9.1 Язык программирования «Python»

9.1 История создания

Язык программирования Python был задуман в 1980-х годах, а его создание началось в декабре 1989 года Гвидо ван Россумом в составе центра математики и информатики в Нидерландах. Язык Python был задуман как потомок языка программирования ABC, способный к обработке исключений и взаимодействию с операционной системой Амёба. Ван Россум является основным автором Python-а и по сей день продолжает выполнять центральную роль в принятии решений относительно развития языка.

Версия Python 2.0 была выпущена 16 октября 2000 года и включала в себя много новых крупных функций – таких как полный сборщик мусора и поддержка Unicode. Однако наиболее важным из всех изменений было изменение самого процесса развития языка и переход на более прозрачный процесс его создания.

9.2 Основные особенности языка

В отличие от других языков программирования, Python не только распространяется совершенно бесплатно, он не имеет абсолютно никаких ограничений в условиях применения. Никто не ограничивает коммерческое использование программных продуктов, написанных на этом языке, без каких-либо лицензионных отчислений. Программисты также вольны модернизировать язык, не ставя в известность автора.

Версия 1.0

Python 1.0 появился в январе 1994 года. Основными новыми возможностями, включенными в этот релиз, были средства функционального программирования: лямбда-исчисление, map, filter и свёртка списка. Ван Россум утверждал, что «Python приобрёл lambda, reduce(), filter() и map() благодаря любителю Lisp, которому их не хватало, и он предоставил патчи, реализующие эти функции».

Последней версией, выпущенной Ван Россумом во время работы в центре математики и информатики, был Python 1.2. С 1995 года Ван Россум продолжил работу над Python-ом в корпорации национальных исследовательских инициатив в городе Рестон, штат Вирджиния, где было выпущено несколько версий языка.

К версии 1.4 Python включал в себя множество новых функций, среди которых наиболее заметными были позаимствованные в Modula-3 именованные параметры и встроенная поддержка комплексных чисел. Также в 1.4 появилась простая форма сокрытия данных при помощи name mangling.

Версия BeOpen

В 2000 году ядро команды разработчиков Python перешло в BeOpen.com, сформировав команду BeOpen PythonLab. Python 2.0 был единственным релизом BeOpen.com. После него Ван Россум и остальные разработчики PythonLab присоединились к Digital Creations.

Версия 2.0

В версии Python 2.0 появилось списковое включение – функция, заимствованная из функциональных языков программирования SETL и Haskell. Синтаксис в Python для этой конструкции очень похож на Haskell, за исключением того, что в Haskell предпочли использовать символы пунктуации, а в Python — ключевые слова. Также в Python 2.0 была добавлена система сборки мусора с поддержкой циклических ссылок.

Начиная с альфа релиза Python 2.1 весь код, техническая документация и спецификации принадлежат некоммерческой организации Python Software Foundation (PSF), созданной в 2001 году по образцу Apache Software Foundation. Релиз включал изменение в спецификацию языка, поддерживающее вложенные области видимости, как в языках со статической (лексической) областью видимости.

В Python 2.2 было объединение базовых типов Python и классов, создаваемых пользователем, в одной иерархии. Это сделало Python полностью объектно-ориентированным языком.

Версия 3.0

Python 3.0 (называемый также «Python 3000» или «Py3K») разрабатывался с целью устранения фундаментальных изъянов в языке. Эти изменения не могли быть сделаны при условии сохранения полной обратной совместимости с 2.x версией, поэтому потребовалось изменение главного номера версии. Ведущим принципом разработки Python 3 было: «уменьшение дублирующейся функциональности устранением устаревших способов сделать это». Python 3.0 был выпущен 3 декабря 2008 года.

Первая версии Python 3.0 была выпущена 3 декабря 2008 года после длительного периода тестирования. Многие функции в этой новой версии были совместимы с Python 2.6 и Python 2.7.

9.3 Область применения

Python активно использует такие гиганты как:

- Google, использует Python в своей поисковой системе.
- Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm и IBM, используют Python для тестирования аппаратного обеспечения.
- NSA для шифрования и анализа разведданных.
- YouTube в которой служба коллективного использования видеоматериалов в значительной степени реализована на Python.
- JPMorgan Chase, UBS, Getco и Citadel применяют Python для прогнозирования финансового рынка.

Популярная программа BitTorrent для обмена файлами в пиринговых сетях написана на языке Python.

Популярный веб-фреймворк App Engine от компании Google использует Python в качестве прикладного языка программирования.

NASA, Los Alamos, JPL и Fermilab используют Python для научных вычислений.

Заключение

Изобретение языков программирования высшего уровня, а также их постоянное совершенствование и развитие, позволило человеку не только общаться с машиной и понимать ее, но использовать ЭВМ для сложнейших расчетов в области самолетостроения, ракетостроения, медицины и даже экономики.

На сегодняшний день, любое среднее и крупное предприятие, имеет в своем штате группу программистов, обладающими знаниями программирования различными языками, которые редактируют, изменяют, и модифицируют программы используемыми сотрудниками предприятия. Это говорит о том, что на рынке труда пользуются спросом обладающими знаниями и опытом работы с различными языками программирования.

В данной курсовой работе, нами были рассмотрены самые распространенные языки программирования, такие как: линейка языков «С», Pascal, Delphi, Java, Java Script, Python которые используется для научных вычислений, для обучения программированию начинающих программистов и решение обширного круга задач в IT индустрии.

Список использованной литературы

1. Курносов А.П., Кулев С.А., Улезько А.В. и др.; Информатика/ Под ред. А.П. Курносова.-М.: КолосС, 2005г;
2. Островский В.А. Информатика: учебный курс для вузов. М.: Высшая школа, 2000г;
3. Суханов М. В., Бачурин И. В., Майоров И. С. Основы Microsoft .NET Framework и языка программирования C#: учебное пособие. Издательство: Национальный Открытый Университет «ИНТУИТ», 2016г;
4. Сузи Р. А. Язык программирования Python: курс Издательство: Интернет-Университет Информационных Технологий, 2007г;
5. Фридман А. Л. Язык программирования Си++. Издательство: Интернет-Университет Информационных Технологий, 2004г;

6. Баженова И. Ю. Язык программирования Java. Издательство: Диалог-МИФИ, 2008г;
7. Х.М.Дейтел, П.Дж.Дейтел. Как программировать на C++ 5-ое издание 2008г;
8. http://progopedia.ru/language/ada/
9. http://www.ada-ru.org
10. http://www.visual.2000.ru
11. http://cybern.ru
12. http://vbbook.ru
13. http://java-study.ru
14. http://opensourcerules.net
15. https://evmhistory.ru
16. http://library.bmstu.ru
17. https://appsstudio.ru
18. https://it-black.ru