Содержание:

ВВЕДЕНИЕ

Жизнь во время активного развития и повсеместного внедрения новых информационных технологий предоставляет множество преимуществ, но также ужесточает и требования к нам всем. В более жестких условиях конкуренции оказываются не только сами люди, как личности, сотрудники, руководители, но и целые организации и общественные институты. По–настоящему современной компанией сегодня не может считаться та, которая не оборудована по крайней мере минимальным «информационным набором» – простейшей автоматизированной информационной системой. Такая система может использоваться как для автоматизации большинства процессов в компании, так и для выполнения лишь отдельных операций.

Сегодня среди автоматизированных информационных систем плотно укоренились так называемые экономические информационные системы. По сущности это та же информационная система, предназначенная для сбора, хранения и обработки информации. Отличие заключается в том, что система такого типа предназначена для работы с конкретным экономическим объектом, в частности – компанией, производством, учреждением и т.д. Экономические автоматизированные системы производятся серийно и могут устанавливаться в компании в виде уже готового продукта. Кроме того, нередко и индивидуальное проектирование, когда такой проект готовится специально для конкретной организации.

Во второй ситуации сегодня на помощь приходят объектно–ориентированные языки программирования, предоставляющие множество возможностей для создания современных удобных программ с пользовательским интерфейсом. Среди современных языков выделяются несколько флагманов, держащихся на вершине спроса последние годы.

Таким образом цель работы – осуществить анализ и оценку средств реализации объектно–ориентированного подхода к проектированию экономической информационной системы.

Для достижения данной цели выполним следующие задачи:

- рассмотреть сущность объектно–ориентированного подхода;
- охарактеризовать основные категории объектно–ориентированного подхода;
- осуществить анализ и оценку языков С ++, Python, Java, Delphi;
- подвести итоги работы.

Объектом данного исследования выступит объектно–ориентированный подход к проектированию информационных систем, а предметом – непосредственно объектно–ориентированные языки программирования, в частности – С ++, Python, Java.

Структура работы содержит две главы, по два и четыре параграфа соответственно. Кроме того, включены такие структурные элементы как введение, заключение и список использованных источников. В списке источников приведены работы, являющиеся теоретической базой данного исследования – это труды известных ученых, осуществляющих научные изыскания относительно процесса программирования, объектно–ориентированного подхода, конкретных языков программирования и т.д.

1. Общая характеристика объектно–ориентированного подхода к программированию

1.1. Сущность объектно–ориентированного подхода

Языки программирования – это искусственно созданные языки. От естественных они отличаются ограниченным числом «слов» и очень строгими правилами записи команд (операторов). Совокупность подобных требований образует синтаксис языка программирования, а смысл каждой команды и других конструкций языка – его семантику.

Языки программирования – это формальные языки общения человека с ЭВМ, предназначенные для описания совокупности инструкций, выполнение которых обеспечивает правильное решение требуемой задачи. Их основная роль заключается в планировании действий по обработке информации. Любой язык

программирования основан на системе понятий, и уже с ее помощью человек может выражать свои соображения.

Связь между языком, на котором мы думаем/программируем, и задачами и решениями, которые мы можем представлять в своем воображении, очень близка. По этой причине ограничивать свойства языка только целями исключения ошибок программиста в лучшем случае опасно. Как и в случае с естественными языками, есть огромная польза быть по крайней мере двуязычным. Язык предоставляет программисту набор концептуальных инструментов, если они не отвечают задаче, то их просто игнорируют. Например, серьезные ограничения концепции указателя заставляют программиста применять вектора и целую арифметику, чтобы реализовать структуры, указатели и т. п. Хорошее проектирование и отсутствие ошибок не может гарантироваться чисто за счет языковых средств

В основе того или иного языка программирования лежит некоторая руководящая идея, оказывающая существенное влияние на стиль соответствующих программ.

Исторически первой была идея процедурного структурирования программ, в соответствии с которой программист должен был решить, какие именно процедуры он будет использовать в своей программе, а затем выбрать наилучшие алгоритмы для реализации этих процедур. Появление этой идеи было следствием недостаточной изученности алгоритмической стороны вычислительных процессов, столь характерной для ранних программных разработок (сороковые – пятидесятые годы). Типичным примером процедурно–ориентированного языка является Фортран – первый и все еще один из наиболее популярных языков программирования. Последовательное использование идеи процедурного структурирования программ привело к созданию обширных библиотек программирования, содержащих множество сравнительно небольших процедур, из которых, как из кирпичиков, можно строить «здание» программы [14].

По мере прогресса в области вычислительной математики акцент в программировании стал смещаться с процедур в сторону организации данных. Оказалось, что эффективная разработка сложных программ нуждается в действенных способах контроля правильности использования данных. Контроль должен осуществляться как на стадии компиляции, так и при прогоне программ, в противном случае, как показала практика, резко возрастают трудности создания крупных программных проектов. Отчетливое осознание этой проблемы привело к созданию Алгола–60, а позже – Паскаля, Модулы–2, Си и множества других языков программирования, имеющих более или менее развитые структуры типов данных.

Логическим следствием развития этого направления стал модульный подход к разработке программ, характеризующийся стремлением «спрятать» данные и процедуры внутри модуля.

Начиная с языка Симула–67, в программировании наметился новый подход, который получил название объектно–ориентированного программирования (ООП). Его руководящая идея заключается в стремлении связать данные с обрабатывающими эти данные процедурами в единое целое – объект. Характерной чертой объектов является инкапсуляция (объединение) данных и алгоритмов их обработки, в результате чего и данные, и процедуры во многом теряют самостоятельное значение. Фактически объектно–ориентированное программирование можно рассматривать как модульное программирование нового уровня, когда вместо во многом случайного, механического объединения процедур и данных акцент делается на их смысловую связь.

Объектно–ориентированный подход использует объектную декомпозицию, при этом статическая структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами. Каждый объект системы обладает своим собственным поведением, моделирующим поведение объекта реального мира [14].

Понятие «объект» впервые было использовано около 30 лет назад в технических средствах при попытках отойти от традиционной архитектуры фон Неймана и преодолеть барьер между высоким уровнем программных абстракций и низким уровнем абстрагирования на уровне компьютеров. С объектно–ориентированной архитектурой также тесно связаны объектно–ориентированные операционные системы. Однако наиболее значительный вклад в объектный подход был внесен объектными и объектно–ориентированными языками программирования [14].

1.2. Основные категории объектно–ориентированного подхода

Концептуальной основой объектно–ориентированного подхода является объектная модель. Основными ее элементами являются:

- абстрагирование;
- инкапсуляция;
- модульность;

- иерархия.

Кроме основных имеются еще три дополнительных элемента, не являющихся в отличие от основных строго обязательными:

- типизация;
- параллелизм;
- устойчивость [6].

Абстрагирование – это выделение существенных характеристик некоторого объекта, которые отличают его от всех других видов объектов и, таким образом, четко определяют его концептуальные границы относительно дальнейшего рассмотрения и анализа. Абстрагирование концентрирует внимание на внешних особенностях объекта и позволяет отделить самые существенные особенности его поведения от деталей их реализации. Выбор правильного набора абстракций для заданной предметной области представляет собой главную задачу объектно–ориентированного проектирования.

Инкапсуляция – это процесс отделения друг от друга отдельных элементов объекта, определяющих его устройство и поведение. Инкапсуляция служит для того, чтобы изолировать интерфейс объекта, отражающий его внешнее поведение, от внутренней реализации объекта. Объектный подход предполагает, что собственные ресурсы, которыми могут манипулировать только методы самого класса, скрыты от внешней среды. Абстрагирование и инкапсуляция являются взаимодополняющими операциями: абстрагирование фокусирует внимание на внешних особенностях объекта, а инкапсуляция (или, иначе, ограничение доступа) не позволяет объектам–пользователям различать внутреннее устройство объекта [21].

Модульность – это свойство системы, связанное с возможностью ее декомпозиции на ряд внутренне связных, но слабо связанных между собой модулей. Инкапсуляция и модульность создают барьеры между абстракциями.

Иерархия – это ранжирование или упорядочение системы абстракций, расположение их по уровням. Основными видами иерархических структур применительно к сложным системам являются структура классов (иерархия по номенклатуре) и структура объектов (иерархия по составу). Примерами иерархии классов являются простое и множественное наследование (один класс использует структурную или функциональную часть соответственно одного класса или нескольких других классов), а примером иерархии объектов являются агрегация.

Типизация – это ограничение, накладываемое на класс объектов и препятствующее взаимозаменяемости различных классов (или сильно сужающее ее возможность). Типизация позволяет защититься от использования объектов одного класса вместо объектов другого или, по крайней мере, управлять таким использованием.

Параллелизм – свойство объектов находиться в активном или пассивном состоянии и различать активные и пассивные объекты между собой.

Устойчивость – свойство объекта существовать во времени (вне зависимости от процесса, породившего данный объект) и (или) в пространстве (при перемещении объекта из адресного пространства, в котором он был создан) [9].

Основные понятия объектно–ориентированного подхода: объект и класс.

Объект определяется как осязаемая реальность – предмет или явление, имеющие четко определяемое поведение. Объект обладает состоянием, поведением и индивидуальностью. Структура и поведение схожих объектов определяют общий для них класс. Термины «экземпляр класса» и «объект» являются эквивалентными. Состояние объекта характеризуется перечнем всех возможных (статических) свойств данного объекта и текущими значениями (динамическими) каждого из этих свойств. Поведение характеризует воздействие объекта на другие объекты и, наоборот, относительно изменения состояния этих объектов и передачи сообщений. Иначе говоря, поведение объекта полностью определяется его действиями. Индивидуальность – это свойства объекта, отличающие его от всех других объектов.

Определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию называется операцией. Как правило, в объектных и объектно–ориентированных языках операции, выполняемые над данным объектом, называются методами и являются составной частью определения класса.

Класс – это множество объектов, связанных общностью структуры и поведения. Любой объект является экземпляром класса. Определение классов и объектов – одна из самых сложных задач объектно–ориентированного проектирования [9].

Следующую группу важных понятий объектного подхода составляют наследование и полиморфизм. Понятие полиморфизма может быть интерпретировано, как способность класса принадлежать более чем одному типу Наследование означает построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов.

Объектно–ориентированная система изначально строится с учетом ее эволюции. Наследование и полиморфизм обеспечивают возможность определения новой функциональности классов с помощью создания производных классов – потомков базовых классов. Потомки наследуют характеристики родительских классов без изменения их первоначального описания и добавляют при необходимости собственные структуры данных и методы. Определение производных классов, при котором задаются только различия или уточнения, в огромной степени экономит время и усилия при производстве и использовании спецификаций и программного кода.

Важным качеством объектного подхода является согласованность моделей деятельности организации и моделей проектируемой системы от стадии формирования требований до стадии реализации. Требование согласованности моделей выполняется благодаря возможности применения абстрагирования, модульности, полиморфизма на всех стадиях разработки. Модели ранних стадий могут быть непосредственно подвергнуты сравнению с моделями реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы [22].

В первой главе работы рассмотрены основные понятия и категории объектно–ориентированного программирования. Рассмотрение данных вопросов позволит наиболее эффективно осуществить анализ и оценку средств реализации объектно–ориентированного подхода к созданию экономических информационных систем.

2. Средства реализации объектно–ориентированного подхода к проектированию экономической информационной системы

2.1 Java

Java – это язык программирования общего назначения, который следует парадигме объектно–ориентированного программирования и подходу «Написать один раз и

использовать везде». Java используется для настольных, сетевых, мобильных и корпоративных приложений.

Java – это не только язык программирования, но и экосистема инструментов, охватывающая почти все, что может понадобиться при программировании на Java. В нее входят:

Java Development Kit (JDK) – комплект разработчика Java. С помощью JDK и стандартного блокнота можно писать и запускать/ компилировать код на Java;

Java Runtime Environment (JRE) – исполняющая система Java. Механизм распространения программного обеспечения, состоит из автономной виртуальной машины Java, стандартной библиотеки Java и инструментов настройки.

Integrated Development Environment (IDE) – интегрированная среда разработки. Инструменты, которые помогают запускать, редактировать и компилировать код. Самые популярные из них – IntelliJ IDEA, Eclipse и NetBeans [24].

Java можно найти везде. Это основной язык разработки для Android. Он используется в веб–приложениях, правительственных веб–сайтах и технологиях обработки больших данных, таких как Hadoop и Apache Storm. Java подходит и для научных проектов, особенно в области обработки естественного языка. Язык Java преобладал и в программировании для мобильных устройств, задолго до появления смартфонов – первые мобильные игры в начале 2000–х годов были написаны на Java. Java, благодаря своей долгой истории, заработал свое место в Зале славы программирования. Индекс TIOBE, один из самых авторитетных индексов популярности программ в мире, при составлении рейтинга использует результаты поисковой выдачи. Несмотря на растущую популярность Go и Python, Java остается на вершине списка уже более десятилетия.

Java внес в мир программирования новые принципы:

1. Гибкость. Java доказала, что C – процедурный, управляемый вручную и зависящий от платформы код – это не предел совершенства . Благодаря Java, все больше людей начали применять объектно–ориентированное программирование, которое сейчас используется повсеместно.
2. Апплеты. Еще до появления JavaScript, в Java добавили апплеты – небольшие веб–программы, которые предоставляют интерактивные элементы для визуализации и обучения. Они не используются ни для чего, кроме простой анимации, однако апплеты привлекли внимание многих программистов и

подтолкнули их к разработке HTML5, Flash и JavaScript.

3. Разработка через тестирование. Java TDD – уже давно не экспериментальная практика, а стандартный способ разработки программного обеспечения. Введение JUnit в 2000 году считается одним из самых больших достижений Java [7].

В Java есть 5 основных принципов объектно–ориентированного подхода, а именно:

1. Все является объектом

Все данные программы хранятся в объектах. Каждый объект создается (есть средства для создания объектов), существует какое–то время, потом уничтожается.

2. Программа есть группа объектов, общающихся друг с другом

Кроме того, что объект хранит какие–то данные, он умеет выполнять различные операции над своими данными и возвращать результаты этих операций. Теоретически эти операции выполняются как реакция на получение некоторого сообщения данным объектом. Практически это происходит при вызове метода данного объекта.

3. Каждый объект имеет свою память, состоящую из других объектов и/или элементарных данных.

Объект хранит некоторые данные. Эти данные – это другие объекты, входящие в состав данного объекта и/или данные элементарных типов, такие как целое, вещественное, символ, и т.п.

4. Каждый объект имеет свой тип (класс)

Т.е. в объектно–ориентированном подходе не рассматривается возможность создания произвольного объекта, состоящего из того, например, что мы укажем в момент его создания. Все объекты строго типизированы. Мы должны сначала описать (создать) тип (класс) объекта, указав в этом описании из каких элементов (полей) будут состоять объекты данного типа. После этого мы можем создавать объекты этого типа. Все они будут состоять из одних и тех же элементов (полей).

5. Все объекты одного и того же типа могут получать одни и те же сообщения [12]

Кроме описания структуры данных, входящих в объекты данного типа, описание типа содержит описание всех сообщений, которые могут получать объекты данного типа (всех методов данного класса). Более того, в описании типа мы должны задать не только перечень и сигнатуру сообщений данного типа, но и алгоритмы их обработки.

1. Достоинства:

- наибольшая среди всех языков программирования степень переносимости программ;
- мощные стандартные библиотеки;
- встроенная поддержка работы в сетях (как локальных, так и internet/intranet).

2. Недостатки:

- низкое, в сравнении с другими языками, быстродействие, повышенные требования к объему оперативной памяти (ОП);
- большой объем стандартных библиотек и технологий создает сложности в изучении языка;
- постоянное развитие языка вызывает наличие как устаревших, так и новых средств, имеющих одно и то же функциональное назначение [13].

Так же некоторые особенности языка:

- Java является полностью объектно–ориентированным языком. Например, C++ тоже является объектно–ориентированным, но в нем есть возможность писать программы не в объектно–ориентированном стиле, а в Java так нельзя;
- реализован с использованием интерпретации P–кода (байт–кода). Т.е. программа сначала транслируется в машинонезависимый P–код, а потом интерпретируется некоторой программой–интерпретатором (виртуальная Java–машина, JVM) [13].

2.2. Python

Python – это универсальный современный язык программирования высокого уровня, к преимуществам которого относят высокую производительность программных решений и структурированный, хорошо читаемый код. Синтаксис Питона максимально облегчен, что позволяет выучить его за сравнительно короткое время. Ядро имеет очень удобную структуру, а широкий перечень встроенных

библиотек позволяет применять внушительный набор полезных функций и возможностей. ЯП может использоваться для написания прикладных приложений, а также разработки WEB–сервисов [18].

Python может поддерживать широкий перечень стилей разработки приложений, в том числе, очень удобен для работы с ООП и функционального программирования. Один из самых популярных интерпретаторов языка – CPython, написанный на Си. Распространяется эта среда разработки бесплатно по свободной лицензии. Интерпретатор поддерживает большинство популярных платформ.

Питон активно развивается. Примерно раз в 2 года выходят обновления. Важной особенностью языка является отсутствие таких стандартов кодировки как ANSI, ISO и некоторых других, они работают благодаря интерпретатору.

Язык начал разрабатываться во второй половине 80–х г.г. прошлого века. Автором Питона стал программист из Нидерландов по имени Гвидо ван Россум. Изначально язык должен был стать объектно–ориентированным. Фактически, это был язык сценариев, т.е. скриптовый язык. В феврале 1991 года ван Россум опубликовал исходный код языка в одной из новостных групп.

Основными факторами успеха Python стали удачный выбор места презентации в популярном и массовом профессиональном сообществе в сочетании с действительно простым кодом и широкими возможностями. Впоследствии Гвидо создал специализированный портал PEP, где идет регулярное обсуждение по развитию и улучшению продукта [18].

В 2008 года появилось большое обновление языка – Python 3.0. Версия продукта известна так же под названием Py3k. В этой версии были устранены многие ключевые недоработки в архитектуре ядра. Что было важно – новая версия продукта сохранила полную совместимость с более старыми вариантами. Сегодня разработчиками поддерживается две линии – Python 3.x и 2.x

Питон – не самый «молодой» язык программирования, но и не слишком старый. К моменту его создания уже существовали такие «монстры», как Pascal или C. А потому при создании ЯП авторы старались взять лучшее из различных платформ для разработчиков. Фактически Python представляет из себя своеобразный «микс» удачных решений более чем из 8 различных языков. К примеру, байт компиляция появилась еще до создания Питона, но была очень удачна в него интегрирована.

Питон поддерживает практически все распространенные операционные системы. Он может прекрасно работать на карманных компьютерах, так и на больших серверах. В случае если платформа значительно устаревает, она исключается из поддержки ядра. К примеру, версии языка, начиная от 2.6, уже не работают с платформами Windows 95, 98 и ME. В случае необходимости можно воспользоваться более старыми версиями, отказавшись от применения современных инструментов языка. И тогда приложение будет работать в том числе с этими ОС. Для старых версий периодически выходят патчи. Язык также может поддерживать работу с виртуальной машиной Java.

Язык программирования имеет четко структурированное семантическое ядро и достаточно простой синтаксис. Все, что пишется на этом языке, всегда легко читаемо [13].

Набор операторов в языке вполне стандартен. Удобная особенность синтаксиса – это форматирование текста кода при помощи разбивки их на блоки с помощью отступов, которые создают нажатием клавиш «Space» и «Tab». В синтаксисе отсутствуют фигурные или операторные скобки, обозначающие начало и конец блока. Такое решение заметно сокращает количество строк тела программы и приучает программиста соблюдать хороший стиль и аккуратность при написании кода.

В 2018 году в Питоне были изменены некоторые ключевые термины, но это скорее упростило понимание. А потому проблем у разработчиков при изучении документации не возникает.

Питон – это высокоуровневый язык, который можно применять и для создания прикладных программ, и для WEB разработки. Производительность платформы весьма высока, код отличается простой и читабельностью.

Иногда его сравнивают с такими популярными платформами как Ruby, но, в отличие от него, Python требует меньше оперативной памяти, быстрее взаимодействует с процессором.

Краткий перечень возможностей:

- любой описанный класс единовременно представляет из себя и объект;
- функция множественного наследования;
- поддержка виртуальных функций;
- возможность легко управлять именами скрывать их особыми метками;

- возможность жизнью объекта и распределение памяти;
- управление работы операторов как символьных, так и логических;
- возможность имитировать поле;
- управление полями – как прямой, так и частичный доступ;
- контроль над самыми распространенными операциями. от глубокого до итерации по объекту;
- возможность создавать триггеры и классы [12].

2.3. C ++

C++ является расширением языка C. C представляет собой гибкий и мощный язык программирования, использовавшийся для разработки наиболее важных программных продуктов в течение прошедших лет. Однако, как только проект превышает определенные размеры, возможности применения языка C достигают своих границ. В зависимости от проекта, программы размером от 25000 до 100000 строк оказываются трудными для разработки и управления потому, что их трудно охватить целиком. Работая в Bell Laboratories в Murray Hill, штат Нью–Джерси, Бьярн Страуструп (Bjarne Stroustrup) добавил к языку C несколько расширений с целью решить эту проблему. Первоначально язык назывался «C с классами». Это название было заменено на C++ в 1983 году [17].

Большинство сделанных Страуструпом добавлений к C поддерживают объектно–ориентированное программирование (далее – ООП), которое иногда сокращенно называют ООП. В следующем разделе будут кратко изложены основные концепции объектно–ориентированного программирования. Как отмечает Страуструп, целый ряд объектно–ориентированных концепций был добавлен в C++, основываясь на языке Симула–67. Поэтому C++ представляет собой смесь двух мощных программных методов.

С момента своего возникновения C++ подвергался серьезным ревизиям трижды, первый раз в 1985 году, второй – в 1989 году. Третий пересмотр языка произошел в связи с работой над стандартом ANSI для C++. Первая версия предложенного стандарта была создана к 25 января 1994 года. Комитет ANSI по языку C++ практически сохранил все черты языка, определенные Страуструпом, и добавил несколько новых. Процесс стандартизации обычно является достаточно медленным, и стандартизация C++ не является исключением.

Изобретая C++ путем добавления к языку C поддержки объектно–ориентированного программирования, Страуструп представлял всю важность сохранения философии языка C, включая его эффективность, гибкость и то, что именно программист, а не язык отвечает за разрабатываемое программное обеспечение. Как будет видно, справиться с этой задачей было нелегко. C++ обеспечивает всю свободу языка C одновременно с мощью объектов. Как отмечал Страуструп, C++ позволяет добиться ясности, расширяемости и легкости сопровождения за счет структуризации причем без потери эффективности [17].

Хотя первоначально C++ был нацелен на работу с очень большими программами, это не ограничивает его применение. Фактически объектно–ориентированные атрибуты языка C++ могут быть эффективно применены фактически к любой задаче программирования. Этот язык часто используется для таких проектов, как создание редакторов, баз данных, персональных систем работы с файлами и коммуникационных программ. Благодаря тому, что C++ унаследовал эффективность языка C, с его помощью разрабатывается высокопроизводительное программное обеспечение.

Поскольку C++ является надмножеством C, то большинство программ на языке C являются также программами и на языке C++. (Имеется несколько небольших различий между C и C++, благодаря которым некоторые типы программ на языке C не будут компилироваться компилятором языка C++. Можно писать программы на C++, которые выглядят в точности как программы на языке C, но в таком случае не будут использоваться преимущества, предоставляемые C++ – программистам. Кроме того, большинство программистов, пишущих на языке C++, используют стиль и некоторые особенности написания программ, которые присущи только C++ [3].

C++ – компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования как процедурное программирование, объектно–ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником – языком C, – наибольшее внимание уделено поддержке объектно–ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ [4].

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 [11].

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.

C++ – язык, складывающийся эволюционно. Каждый элемент C++ заимствовался из других языков отдельно и независимо от остальных элементов (ничто из предложенного C++ за всю историю его развития не было новшеством в Computer Science), что сделало язык чрезвычайно сложным, со множеством дублирующихся и взаимно противоречивых элементов, блоки которых основаны на разных формальных базах [1].

Критики C++ не противопоставляют ему какой–либо конкретный язык, а наоборот, утверждают, что для всякого случая применения C++ всегда существует альтернативный инструментарий, позволяющий решить ту же задачу более эффективно и качественно. В свою очередь, сторонники C++ считают некорректным сравнивать различные аспекты C++ с совершенно различными языками, так как общий набор средств и возможностей C++ существенно шире, чем в большинстве языков, с которыми проводится сравнение, и сама по себе широта возможностей, на их взгляд, является веским оправданием несовершенства каждой отдельно взятой возможности. Более того, по их мнению, высокая совместимость с Си является одной из принципиальных черт языка, и потому все недостатки C++ оправданы преимуществами, предоставляемыми этой совместимостью [1].

Достоинства:

- высокая совместимость с языком C;

- вычислительная производительность;
- поддержка различных стилей программирования: структурное, объектно–ориентированное, обобщённое программирование, функциональное программирование, порождающее метапрограммирование;
- автоматический вызов деструкторов объектов (в порядке обратном вызову конструкторов) упрощает и повышает надёжность управления памятью и другими ресурсами (открытыми файлами, сетевыми соединениями, т. п.);
- перегрузка операторов;
- шаблоны (дают возможность построения обобщённых контейнеров и алгоритмов для разных типов данных);
- возможность расширения языка для поддержки парадигм, которые не поддерживаются компиляторами напрямую;
- доступность. Для C++ существует огромное количество учебной литературы, переведённой на всевозможные языки.

Недостатки:

- плохо продуманный синтаксис сужает спектр применимости языка;
- язык не содержит многих важных возможностей;
- язык содержит опасные возможности;
- производительность труда программистов на языке оказывается неоправданно низка;
- громоздкость синтаксиса;
- тяжелое наследие;
- необходимость следить за памятью [12].

В целом C++ является достаточно удобным инструментом реализации целей объектно–ориентированного программирования, и в частности – проектирования экономических информационных систем, показывающим высокую эффективность и функциональность готовых программных продуктов.

2.4. Delphi

Концепция Delphi 1 была реализована в конце 1994 года, когда вышла первая версия среды разработки. В основу этого программного продукта легли концепции объектно–ориентированного программирования на базе языка Object Pascal и визуального подхода к построению приложений.

После выхода Delphi 1 все компьютерные издания писали об этой среде, как об «убийце Visual Basic». Появление Delphi 2 (32–разрядной) ознаменовало новую эпоху, – появился доступ к возможностям программных интерфейсов Windows NT и Windows 95. Delphi 2 стала средством разработки полноценных приложений клиент/сервер. Вскоре Delphi 3 предоставила разработчикам средства создания распределенных многоуровневых приложений и полноценный инструментарий проектирования приложений для Internet и Intranet. Появилась полноценная поддержка com – модели объектов, ставшей краеугольным камнем современного программирования. Четвертая версия Delphi позволяет полностью интегрировать ваши разработки с объектами com. Поддержка архитектуры corba (common object request broker architecture) открывает перед приложениями, созданными в delphi для платформы wintel (windows + intel), мир других операционных систем (unix, os/2, wms).

Delphi представляет следующие новые свойства и усовершенствования:

- Новые расширения языка. В Delphi в язык Object Pascal включены динамические массивы, методы обработки переполнения, установка значения параметров по умолчанию, и многое другое;
- Менеджер Проекта. Новый менеджер проекта позволяет Вам объединять проекты, которые работают вместе в одну проектную группу. Это позволяет организовать как работу взаимозависимых проектов, таких как однозадачные и многозадачные приложения или dll, так и совместную работу исполняемых программ;
- Новый проводник. Новый проводник содержит выполняемые классы, навигацию по модулям, и браузер кода. Проводник кода делает создание классов проще. Также проводник позволяет быстро перемещаться через файлы модуля, а так же между интерфейсом и реализацией;
- Закрепляемые окна инструментов. IDE (Интегрированная Среда разработки) содержит более перенастраиваемую конфигурацию окон инструментов, которые можно закреплять с редактором кода;
    - Улучшенная отладка. Интегрированный отладчик имеет много новых свойств, включая удаленную и многопроцессорную отладку, просмотр кода центрального процессора, инспекторов, усовершенствованные точки прерывания, отладчик специфических подменю и закрепленных окон;
- Усовершенствования Activex;

- Усовершенствования VCL. Иерархия объектов Delphi была расширена, чтобы включить новый компонент для Nt Service приложений. Кроме того, новый компонент выполняемого списка (на Стандартной странице палитры), позволяет централизовать управление меню и команд от кнопок. Управление VCL расширено, чтобы поддерживать drag–and–drop перетаскивания, обеспечивать дополнительный контроль над размещением окна, и многое другое.

Delphi – это комбинация нескольких важнейших технологий:

- высокопроизводительный компилятор в машинный код;
- объектно–ориентированная модель компонент;
- визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов;
- масштабируемые средства для построения баз данных.

Компилятор, встроенный в Delphi, обеспечивает высокую производительность, необходимую для построения приложений в архитектуре «клиент–сервер». Он предлагает легкость разработки и быстрое время проверки готового программного блока, характерного для языков четвертого поколения. Кроме того, Delphi обеспечивает быструю разработку без необходимости писать вставки на Си или ручного написания кода (хотя это возможно).

В процессе построения приложения разработчик выбирает из палитры компонент готовые компоненты как художник, делающий крупные мазки кистью. Еще до компиляции он видит результаты своей работы – после подключения к источнику данных их можно видеть отображенными на форме, можно перемещаться по данным, представлять их в том или ином виде. В этом смысле проектирование в Delphi мало чем отличается от проектирования в интерпретирующей среде, однако после выполнения компиляции мы получаем код, который исполняется в 10–20 раз быстрее, чем то же самое, сделанное при помощи интерпретатора. Кроме того, компилятор компилятору рознь, в Delphi компиляция производится непосредственно в родной машинный код, в то время как существуют компиляторы, превращающие программу в так называемый p–код, который затем интерпретируется виртуальной p–машиной. Это не может не сказаться на фактическом быстродействии готового приложения.

В стандартную поставку Delphi входят основные объекты, которые образуют удачно подобранную иерархию базовых классов. Но если возникнет необходимость

в решении какой–то специфической проблемы на Delphi, то лучше просмотреть список свободно распространяемых или коммерческих компонент, разработанных третьими фирмами, количество этих компонент в настоящее время составляет несколько тысяч. Событийная модель в Windows всегда была сложна для понимания и отладки. Но именно разработка интерфейса в Delphi является самой простой задачей для программиста.

Объекты БД в Delphi основаны на SQL и включают в себя полную мощь Borland Database Engine. В состав Delphi также включен Borland SQL LINK, поэтому доступ к СУБД Oracle, Sybase, Informix и Interbase происходит с высокой эффективностью. Кроме того, Delphi включает в себя локальный сервер Interbase для того, чтобы можно было разработать расширяемые на любые внешние sql–сервера приложения в офлайновом режиме. Разработчик в среде Delphi, проектирующий информационную систему для локальной машины (к примеру, небольшую систему учета медицинских карточек для одного компьютера), может использовать для хранения информации файлы формата .dbf или .db (paradox). Если же он будет использовать локальный interbase for windows (это локальный SQL–сервер, входящий в поставку), то его приложение безо всяких изменений будет работать и в составе большой системы с архитектурой клиент–сервер.

Во второй главе работы рассмотрены популярные сегодня средства реализации объектно–ориентированного программирования. Рассмотренные средства являются достаточно универсальными, что делает их подходящими для использования в процессе конструирования, в том числе, и экономической информационной системы. Анализ достоинств и недостатков рассмотренных языков объектно–ориентированного программирования также позволяют говорить о дальнейшем их развитии и, соответственно, усовершенствовании. Также можно сделать опосредованный вывод о том, что с течением времени и с развитием средств реализации объектно–ориентированного подхода к программированию, эффективность создаваемых с их помощью информационных систем будет возрастать. В вопросе функционирования экономических информационных систем это приобретает особенную важность, так как от их надежности зависит как деятельность отдельных организаций, так и целых отраслей.

В частности, рассмотрены такие языки, как C++, Python, Delphi и Java. Рассмотренные средства реализации объектно–ориентированного подхода обладают различными преимуществами и недостатками, при этом каждое из них может в той или иной степени подходить для реализации определенных целей и задач программирования. Выбор определенного языка программирования должен

осуществляться индивидуально, исходя из целей, возможностей и ресурсов проекта.

Очевидно, что код на JavaScript проще для понимания начинающего разработчика с базовым знанием английского языка, чем код на C++. Это происходит из–за статической типизации C++: в отличие от того же JavaScript, мы не можем в пределах одной переменной перевести число в строку, к тому же метод String(), принимающий один параметр в JS выглядит логичнее, чем метод itoa() в C++, принимающий три параметра.

Все эти сложности делают C++ довольно противоречивым языком. С одной стороны, это низкоуровневый кроссплатформенный язык программирования, который еще долгое время будет актуальным из–за своей универсальности и эффективности. С другой стороны, начинающим разработчиками будет сложнее овладевать различными алгоритмами и принципами программирования на C++ именно из–за его статической типизации, ограничивающей работу с переменными. Тем не менее, начинающим разработчикам стоит начать знакомство с программированием именно с C++, так как они смогут освоить C–подобный синтаксис, а затем – парадигмы объектно–ориентированного подхода к созданию информационных систем, используя при этом универсальный язык программирования, который в будущем поможет им в реализации больших проектов.

ЗАКЛЮЧЕНИЕ

В ходе проведения исследования была достигнута поставленная цель – осуществлен анализ и оценка средств объектно–ориентированного подхода к проектированию экономической информационной системы.

Для достижения цели были выполнены следующие задачи:

- рассмотрена суть объектно–ориентированного подхода;
- охарактеризованы основные категории объектно–ориентированного подхода;
- осуществлен анализ и оценка языков C ++, Python, Java и Delphi.

Обзор наиболее распространенных языков объектно–ориентированного программирования показал, что каждый из них обладает относительно тождественным числом преимуществ и недостатков, что говорит о субъективности

применимости того или иного языка. Так, в зависимости от целей и задач разработки экономической информационной системы, может быть выбран любой из рассмотренных языков программирования. Несмотря на это, есть то, что объединяет все рассмотренные языки объектно–ориентированного программирования – это высокие требования к программисту, его навыкам и знаниям. Так, для продуктивной работы с рассмотренными языками обязательно наличие опыта и навыков работы с более примитивными языками низкого уровня.

Технологии совершенствуются и усложняются буквально каждый день, в связи с чем проблему, изучаемую в настоящем исследовании, нельзя назвать окончательно раскрытой. Текст данной работы может быть дополнен и использован в качестве теоретической базы для дальнейших научных изысканий, которые непременно потребуются с течением времени и появлением и усовершенствованием средств реализации объектно–ориентированного программирования, в частности – для проектирования экономических информационных систем.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) / В. А. Биллиг. – М.: Интернет–университет информационных технологий, Бином. Лаборатория знаний, 2016. – 584 с.
2. Буховец, А. Г. Алгоритмы вычислительной статистики в системе R. Учебное пособие / А. Г. Буховец, П. В. Москалев. – М.: Лань, 2015. – 160 с.
3. Гавриков, М. М. Теоретические основы разработки и реализации языков программирования / М. М. Гавриков, А. Н. Иванченко, Д. В. Гринченков. – М.: КноРус, 2014. – 184 с.
4. Гергель, В. П. Современные языки и технологии параллельного программирования / В. П. Гергель. – М.: Издательство МГУ, 2016. – 408 с.
5. Герман, О. Программирование на Java и C# для студента / О. Герман, Ю. Герман. – М.: БХВ–Петербург, 2014. – 512 с.
6. Зыков, С. В. Введение в теорию программирования. Курс лекций. Учебное пособие / С. В. Зыков. – М.: Интернет–университет информационных технологий, 2017. – 400 с.
7. Ишкова, Э. А. C#. Начала программирования / Э. А. Ишкова. – М.: Бином–Пресс, 2016. – 334 с.
8. Кетков, Ю. Л. Свободное программное обеспечение. FREE PASCAL для студентов и школьников (+ CD) / Ю.Л. Кетков, А.Ю. Кетков. – М.: БХВ–Петербург, 2017. – 376 с.

9. Культин, Н. Visual Basic для студентов и школьников / Н. Культин. – М.: БХВ–Петербург, 2017. – 354 с.
10. Медведик, В. И. Практика программирования на Паскаль. Задачи и решения. Учебное пособие / В. И. Медведик. – М.: ДМК Пресс, 2015. – 590 с.
11. Опалева, Э. А. Языки программирования и методы трансляции / Э. А. Опалева, В. П. Самойленко. – М.: БХВ–Петербург, 2015. – 480 с.
12. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – М.: Питер, 2016. – 464 с.
13. Павловская, Т. А. C/C++. Процедурное и объектно–ориентированное программирование. Учебник / Т. А. Павловская. – М.: Питер, 2015. – 496 с.
14. Рапаков, Г. Г. Turbo Pascal для студентов и школьников / Г. Г. Рапаков, С. Ю. Ржеуцкая. – М.: БХВ–Петербург, 2017. – 352 с.
15. Санников, Е. В. Курс практического программирования в Delphi. Объектно–ориентированное программирование / Е. В. Санников. – М.: Солон–Пресс, 2015. – 188 с.
16. Финогенов, К. Г. Использование языка Ассемблера. Учебное пособие / К. Г. Финогенов. – М.: Горячая линия – Телеком, 2017. – 440 с.
17. Хабибуллин, И. Программирование на языке высокого уровня. C/C++ / И. Хабибуллин. – М.: БХВ–Петербург, 2016. – 512 с.
18. Хорев, П. Б. Объектно–ориентированное программирование с примерами на C#. Учебное пособие / П. Б. Хорев. – М.: Форум, Инфра–М, 2016. – 200 с.
19. Черпаков, И. В. Основы программирования. Учебник и практикум / И. В. Черпаков. – М.: Юрайт, 2016. – 220 с.
20. https://www.java.com/ru/about/