Содержание:

Введение

Актуальность исследования.

Несколько десятилетий программирование быстро развивается. Причем, этот процесс сопровождается совершенствования технологических приемов и разработка на их основе таких средств программирования, которые существенно упростили бы процесс написания приложений. Это привело к тому, что на данный момент создаются очень сложные программные продукты.

Первые программы были организованы очень просто. Они состояли из собственно программы на машинном языке и обрабатываемых данных. Но постепенно все усложнилось. Появились новые языки программирования, и параллельно появились и новые методы и средства программирования.

Появление новых методов программирования является результатом того, что начали разрабатываться все сложнее и сложнее программы.

Поэтому, данная тема актуальна, современному программисту необходимо знать современные методы программирования, для того, что бы уметь написать не только простую программу – решение какой-то математической задачи, но и сложную программу.

Целью курсовой работы является изучение современных методов программирования.

Для достижения данной цели необходимо решение следующих **задач:**

- Дать определения понятия основных понятий по данной теме;
- Раскрыть суть методологии программирования;
- Раскрыть суть модульного программирования;
- Раскрыть сущность структурного программирования;
- Раскрыть сущность объектно-ориентированного программирования;
- Описать основные понятия связанные с объектно-ориентированным программированием.

Объектом исследования данной курсовой работы методы программирования. **Предметом исследования** выступает современные методы программирования.

Работа состоит из введения, трех глав («Глава 1. Основные понятия и определения», «Глава 2. Методы современного программирования», «Глава 3 Объектно-ориентированный подход в разработке программ»), заключения и списка использованной литературы.

Глава 1. Основные понятия и определения

1.1. Основные понятия

Процесс программирования связан с различными понятиями. В данном подпункте опишем основные понятия. Причем эти понятия должны знать программисты-разработчики.

Итак, рассмотрим основные понятия.

Компьютер – это машина, с которой все связано. Компьютер – представляет собой комплекс технических и программных средств, которые предназначены для автоматизации подготовки и решения задач пользователя.

Компьютер предназначен для сбора, обработки и хранения различной информации [1]. Причем следует отметить, что автор данного определения определяет понятие «компьютер» как средство работы с информацией.

Как следует из предыдущего определения, работа компьютера обеспечивается двумя составляющими: техническими средствами (hardware) и программным обеспечением (software).

Аппаратное обеспечение (англ. hardware, HW) – это общее описание любого физического (электронного/механического) компонента компьютерной системы, что состоит из печатной платы, микросхемы или другого вида электроники. Прекрасный пример аппаратного средства это экран. Будь то дисплей компьютера, планшета или смартфона – всё это аппаратная часть.

Аппаратное обеспечение также называют: аппаратные средства, компьютерные комплектующие или просто «железо».

Без аппаратного обеспечения компьютер не может существовать и программное обеспечение не может быть использовано[2].

Как компромисс между требованиями аппаратного обеспечения и лёгкостью описания решения задач, были созданы специальные языки для «общения» человека и машины - языки программирования.

Язык программирования - это средство общения между человеком (пользователем) и компьютером (исполнителем). С помощью языка программирования формируются сообщения для компьютера. Эти сообщения должны быть понятны компьютеру[3].

Все программы пишутся на языках программирования людьми, то есть программистами. Причем для работы компьютера нужны только машинные коды.

Основные составляющие языка программирования является синтаксис и семантика.

Синтаксис языка программирования - это набор формальных правил, определяющих какие последовательности символов являются допустимыми в этом языке без учёта вложенного в них смысла.

Синтаксис языка программирования - это сторона языка программирования, которая описывает структуру программ как наборов символов (обычно говорят - безотносительно к содержанию). Каждый язык программирования имеет синтаксическое описание. Обычно синтаксис языка определяют посредством правил Бэкуса-Наура (формальная система описания синтаксиса, в которой одни синтаксические категории последовательно определяются через другие категории)[4].

Семантика языка программирования, вообще говоря, противопоставляется синтаксису. Синтаксис языка описывает «чистый» язык, в то же время семантика приписывает значения (действия) различным синтаксическим конструкциям (иными словами, это правила истолкования). Имеется несколько подходов к определению семантики языков программирования, причем наиболее широко распространены разновидности следующих трех: операционного, денотационного (математического) и деривационного (аксиоматического). При описании семантики в рамках операционного подхода обычно исполнение конструкций языка программирования интерпретируется с помощью некоторой воображаемой (абстрактной) ЭВМ. Деривационная семантика описывает последствия выполнения конструкций языка с помощью языка логики и задания предусловий и постусловий.

Денотационная семантика оперирует понятиями, типичными для математики (множества, соответствия, а также суждения, утверждения и др.)[5].

1.2. Методология программирования

Программирование - это не только код, спецификации, операционные системы, это еще и методология. Существуют книги, которые не связаны непосредственно ни с одним языком программирования, но помогают чувствовать направление и основные тенденции развития индустрии[6].

Методология программирования – совокупность методов, применимых в жизненном цикле программного обеспечения и объединенных общим философским подходом [7].

Все существующие методологии характеризуется следующем:

• философским подходом или основными принципами. Представляют собой принципы, от которых зависит эффективность всей методологии, обычно можно кратко сформулировать и легко объяснить;

• согласованным множеством моделей методов, которые осуществляют данную методологию;

• концепциями (понятиями), которые позволяют более точно определить методы.

В частном случае, когда методология используется на начальной стадии программирования, то есть на стадии конструирования, она называется парадигмой программирования[8].

Выделяют три пути возникновения методологий. Во-первых, они могут являться выражением практического опыта. Во-вторых, методологии могут происходить от одной из четырёх моделей алгоритма:

- абстрактная машина Тьюринга,
- рекурсивные функции Гильберта и Аккермана,
- лямбда-исчисление Черча,
- нормальные алгоритмы Маркова.

В-третьих, методологии можно объяснить через отображение одной из трёх структур языка моделирования на структуру языка программирования. Составными

частями могут быть структура данных, структура управления и логика. Каждое из девяти отображений определяет либо методологию, либо достаточно серьёзный метод программирования. Например, отображение логика-логика лежит в основе логического программирования.

Далее рассмотрим классификацию.

По ядрам

При подходе к методологии, как имеющей ядро, которое соответствует способу описания алгоритма, и дополнительные особенности, выделяют следующие пять основных ядер методологий:

• Методология императивного программирования

• Методология ООП

• Методология функционального программирования

• Методология логическое программирование

• Методология программирования в ограничениях

Можно заметить, что эти методологии находятся на шкале от навигационных (пошаговое управление исполнением) до спецификационных (определение требований к результату).

По топологической специфике

Специфика (топологическая специфика) – представляет собой способ выбора методов для уточнения ядра методологии. Критерием качества той или иной топологии может могут быть общие затраты на разработку программного обеспечения. Следует отметить, что траты на разработку ПО, зависят среди прочего от ключевых языковых абстракций: абстракции данных, управления и модульности.

По специфике реализации

В соответствии с архитектурой аппаратного обеспечения, реализация может быть централизованной или параллельной. Например, методология (императивного) параллельного программирования, методология логического параллельного программирования.

Кроме того, следует выделить и то, что методология может быть гибридной. Например, наиболее часто имеет место совмещение функционального и логического программирования.

Следует отметить, что еще проводятся исследования и по унификации методологий программирования

Существующие языки программирования могут хорошо поддерживать разные методологии. Но надо это понимать, что некоторый язык вообще нельзя применять с несвойственной ему методологией, а только то, что потребуется затратить больше усилий и ресурсов.

Так же методологии программирования делят по общим затратам на решения задач с разными характеристиками (научные расчёты, финансовые задачи, системы реального времени и т. п.). Масштаб задач и эффективность создаваемого программного обеспечения также являются важными факторами при выборе методологии программирования[9].

На данный момент существует четыре широко известных в настоящее время методологии программирования – императивного, объектно-ориентированного, логического, функционального.

Глава 2. Методы современного программирования

2.1 Модульное программирование

Одной из форм разработки программного обеспечения сегодня является модульное программирование.

Модульное программирование - метод разработки программ по частям. Программный модуль – это любой фрагмент описания процесса, оформляемый как самостоятельный программный продукт, пригодный для использования в других описаниях процесса.

Концепцию модульного программирования можно сформулировать в виде нескольких понятий и положений. Основа концепции модульного программирования модуль, который является продуктом процесса разбиения большой задачи на ряд более мелких функционально самостоятельных подзадач.

Этот процесс называется функциональной декомпозицией задачи. Каждый модуль в функциональной декомпозиции представляет собой «черный ящик» с одним входом и одним выходом. Модули связаны между собой только входными и выходными данными[10].

Понятие модульного программирование можно рассмотреть с двух точек зрения с точки зрения теории и с точки зрения практики. А именно:

С точки зрения теории, модульное программирование - представляет собой процесс разбиение всех имен программы на две части. Открытая часть является доступной за пределами модуля, закрытая часть доступна только внутри модуля. Объявления типов, переменных, а также процедуры и функции могут быть отнесены к любой из двух частей.

С практической точки зрения модульное программирование – представляет собой процесс разделение программы на отдельно компилируемые фрагменты**.**

В качестве модульной структуры программы принято использовать древовидную структуру. При разработке может использоваться метод восходящей разработки или метод нисходящей разработки.

Большей популярностью пользуется метод "сверху вниз". Сущность метода в том, что исходная задача сначала разбивается на относительно независимые друг от друга подзадачи. Если подзадача еще сложная, то она разбивается на более простые и т.д. Данный процесс «деления» продолжается до тех пор, пока полученные подзадачи становятся простыми для каждой из них разрабатывается алгоритм и затем реализуется на каком-либо языке программирования. Такая организация задачи является программным модулем. Таким образом, применение модулей является естественным способом разработки и создания сложных программ. Причем созданный программный модуль в дальнейшем может быть детализирован, и полученные в результате этого подзадачи реализуются в виде процедур и функций. В результате, процедуры и функции являются инструментом пошаговой детализации при разработке программ на нижнем, более детальном уровне, а модули - на более верхнем[11]**.**

В концепцию модульного программирования входят следующие положения:

1) каждый модуль выполняет узкоспециализированную независимую функцию;

2) каждый модуль имеет единственную точку входа/выхода;

3) чем меньше размер модуля, тем лучше;

4) каждый модуль может быть создан отдельными программистами бригады и по возможности отдельно отлажен;

5) вся программа создается из модулей.

Исходя, из выше описанных пяти концепция, можно заключить, что, при модульном подходе весь алгоритм условно делится на несколько частей. Причем каждую часть можно проектировать независимо от других. Размер каждого модуля должен быть небольшим. Так как это влияет на его тестирование, то есть, что бы выполнить тестирование очень тщательно. После перевода модулей на конкретный язык программирования и их тестирования производится сборка всех модулей и отладка программы в целом.

Так как концепция модульного программирования как способа создания больших программ появилась достаточно давно. Но следует отметить и то, что большинство языков программирования не позволяют реализовать модуль в точном соответствии с приведенными выше концепциям. В них чаще оперируют с такими понятиями как программный блок и подпрограмма.

Программный блок – представляет собой совокупность описаний (меток, переменных, подпрограмм), определений (констант, типов) и следующая за ними последовательность операторов.

Подпрограмма - представляет собой часть программы, которая оформленная в виде отдельной синтаксической конструкции и имеющая свое имя. Вызов подпрограммы производится посредством указания имени подпрограммы.

В связи с этим, иногда говорят о блочном программировании, концепция которого практически идентична модульному.

Большинство языков программирования позволяют выполнить модуль в виде подпрограммы. Подпрограмма не является самостоятельной программной единицей, поэтому она транслируется, а, следовательно, и отлаживается вместе с вызывающей ее основной программой[12].

Модульное программирование в языке программирования C++ поддерживается с помощью директив препроцессора, пространств имен, классов памяти, исключений и раздельной компиляции (строго говоря, раздельная компиляция не является элементом языка, а относится к его реализации)[13].

Далее рассмотрим, какие **преимущества** и **недостатки** дает модульное программирование.

Преимущества

– возможность разрабатывать модули с использованием различных языков программирования;

– модуль является естественной единицей локализации имен;

– локализация места ошибки: обычно исправление ошибки внутри одного модуля не влечет за собой исправление в других модулях (естественно, если это свойство будет выполняться только при хорошей разбивке программы на части, с малым числом связей между модулями);

– возможность повторного использования разработанных модулей в других программах.

Недостатки:

– модули не являются совсем уж независимыми друг от друга: между ними существуют связи, то есть один модуль иногда может завязываться на переменные, константы и программный код из другого модуля;

– перед тем, как будет произведен первый запуск программы, необходимо собрать все нужные модули для программы в особую сборку. Этот процесс достаточно сложен, так как кроме собственно объединения всех модулей в одну программу, необходимо проконтролировать и установить все связи между этими частями;

– программа-компилятор, с помощью которой делается сборка программы, не видит сразу всех модулей и не может проконтролировать, установлены ли правильно связи между модулями[14].

Несмотря на описанные недостатки, преимущества модульного программирования настолько огромны, что сейчас оно является основным способом разработки сложного программного обеспечения[15].

2.2 Структурное программирование

Следующий метод разработки программ – это структурный метод.

Структурное программирование – это методология программирования, которая основана на системном подходе к анализу, проектированию и реализации программного обеспечения. Если вернуться в историю, то началом использования данной методологии является 70-е года прошлого века в компании IBM, в их разработке участвовали известные ученые Э. Дейкстра, Х. Милс, Э. Кнут, С. Хоор[16]
.

Данная методология и на данный момент пользуется популярностью, а следовательно с ее помощью разрабатываются программные продукты.

Основу структурного программирования составляет концепция модульного программирования, которая описана в предыдущем подпункте данной главы курсовой работы.

Сущность структурного программирования в том, что программы разрабатываются в виде иерархической структуры блоков.

Согласно парадигмы структурного программирования:

1. Любая программа с точки зрения структурного программирования представляет собой структуру, которая построена из трех типов базовых конструкция:

о последовательное исполнение – это есть однократное выполнение операций в том порядке, в котором они записаны в тексте программы,

о ветвление - однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;

о цикл - многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла).

2. Повторяющиеся фрагменты программы (либо не повторяющиеся, но представляющие собой логически целостные вычислительные блоки) могут оформляться в виде т. и. подпрограмм (процедур или функций). В этом случае в тексте основной программы, вместо помещенного в подпрограмму фрагмента, вставляется инструкция вызова подпрограммы. При выполнении такой инструкции выполняется вызванная подпрограмма, после чего исполнение программы продолжается с инструкции, следующей за командой вызова подпрограммы.

3. Разработка программы ведется пошагово, причем методом «снизу-вверх».

То есть, сущность структурного программирования состоит в том, что используя небольшой набор простых управляющих структур и структур данных можно создать программу. Программа строится путем вложений операторов одного в другой.

Структурное программирование делает текст программы более понятным – алгоритм решения ясно виден из исходного текста. Структурное программирование называют программированием без GOTO. Его методология основана на использовании подпрограмм и независимых структур данных, объединяющих связанные между собой совокупности данных. Подпрограммы позволяют заменять в тексте программ упорядоченные блоки команд, отчего программный код становится более компактным. Структурный подход обеспечивает создание более понятных и легко читаемых программ, упрощает их тестирование и отладку[17].

Разработка программы в структурном программировании ведётся пошагово, методом «сверху вниз».

Это позволяет вместо работающих подпрограмм использовать "заглушку", чтобы протестировать работоспособность всей программы в целом. После первого тестирования на работоспособность заглушку заменяют реальной подпрограммой.

Ярким представителем структурного программирования является язык программирования СИ[18].

Следует отметить следующие достоинства структурного программирования:

– повышается надежность программ, за счет хорошего структурирования при проектировании, программа легко поддается тестированию и нет проблем при отладке;

– повышается эффективность программ (структурирование программы позволяет легко находить и корректировать ошибки, а отдельные подпрограммы можно переделывать (модифицировать) независимо от других);

– уменьшается время и стоимость программной разработки;

– улучшается читабельность программ.

Таким образом, технология структурного программирования при разработке серьезных программных комплексов, основана на следующих принципах:

– программирование должно осуществляться сверху вниз;

– весь проект должен быть разбит на модули (подпрограммы) с одним входом и одним выходом;

– подпрограмма должна допускать только три основные структуры – последовательное выполнение, ветвление (if, case) и повторение (for, while, repeat).

– недопустим оператор передачи управления в любую точку программы (goto);

– документация должна создаваться одновременно с программированием в виде комментариев к программе.

Структурное программирование эффективно используется для решения различных математических задач, имеющих алгоритмический характер[19].

Структурное программирование как правило используется для разработки крупных программных комплексом, когда на первый план выходят следующие моменты:

- легкость повторного использования фрагментов кода, оформленных как процедуры (например, в библиотеках);
- легкость прослеживания логики программы;
- возможность сопровождения программного продукта через длительное время после написания кода или кем-то, кроме его разработчика[20].

Программа, которая написана с применением методов структурного программирования понятна, более надежна и облегчает сопровождение.

Глава 3 Объектно-ориентированный подход в разработке программ

3.1 Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП)- это парадигма (совокупность понятий и идей) программирования, в рамках которой «во главу угла» ставят понятия объектов и классов. Сейчас ООП так или иначе присутствует во всех языках, поэтому понимание его основ просто необходимо для всех, кто собирается заняться программированием. Стоит сразу определить базовые понятия класса и

объекта:

Класс - это шаблон, описание ещё не созданного объекта. Класс содержит данные, которые описывают строение объекта и его возможности, методы работы с ним;

Объект - экземпляр класса. То, что «рождено» по «чертежу», то есть по описанию из класса. В качестве примера объекта и класса можно привести технический чертёж для изготовления детали - это класс. Выточенная же на станке по размерам и указаниям из чертежа деталь - объект.

Появление объектно-ориентированного программирования стало результатом возросших требований к функционалу программ, когда описывать объект приходилось раз за разом в разных участках кода. Тогда и было введено понятие класса, параметры которого задавались единожды, а после в коде оставлялись только ссылки на класс, чтобы код самостоятельно «собрал» объект[21].

В качестве более живого примера объектно-ориентированного программирования можно привести мастерскую, где есть старший слесарь (программист) и ученики (разные участки кода). При устаревших парадигмах программирования старшему слесарю пришлось бы сначала объяснить ученику №0, как вырезать деталь (создать объект), затем ученику №1 то же самое, потом ученику №2 и так далее. ООП же даёт слесарю целую пачку бесконечных подробных чертежей (классов) деталей, которые он может раздавать ученикам вместо повторного объяснения (заведения описания объекта). Как нетрудно догадаться, это ускоряет работу и позволяет старшему слесарю уделить внимание более важным проблемам мастерской.

3.2 Основные понятия объектно-ориентированного программирования

Объектно-ориентированное программирование выделяется не только описанной выше системой классов, вернее, её особенность не только в сохранении большого количества параметров. При упущении одного параметра, код приходилось перерывать от и до в поисках ошибки. Поэтому был разработан поведенческий аспект, что означало, что отныне классы могут не только служить вместилищем для данных, но и сами могли бы работать с ними: загружать, сохранять, изменять и выполнять другие операции.

Объектно-ориентированное программирование развивается уже год два, а несколько десятков. На данный момент имеется несколько школ. Каждая из школы предлагает свой набор принципов работы с объектами и по-своему излагает эти принципы. Бурные обсуждения и дискуссии, проходившие между представителями этих школ, позволили создать несколько общепринятых принципов, признанных всеми школами и внедренных во вес объектно-ориентированные языки программирования[22].

В объектно-ориентированном программировании выделяют 4 основных принципа: абстракция, инкапсуляция, наследование и полиморфизм.

Понятие абстракции в объектно-ориентированном программировании

По мнению, Г.Г. Рапакова и С.Ю. Ржеуцкого принцип абстракции является основой идеологии объектно-ориентированного программирования (ООП)[23].

Абстракция - способ выделения самых значимых характеристик объекта, при этом менее значимые отбрасываются. В ООП абстракция - работа только со значимыми характеристиками. Суть этого принципа в том, чтобы отделить составные объекты, состоящие из «меньших» объектов, от этих самых объектов, то есть от их составляющих. Такой подход позволяет работать непосредственно с объектом, не вдаваясь в подробности, из чего же он состоит и как работает. Возвращаясь к примеру про слесарную мастерскую, принцип абстракции заключается в том, что старший слесарь не тратит своё время и ресурсы на определение, из чего ученик сделал деталь, а просто использует её по назначению.

Преимущества абстракции следующие:

- Применяя абстракцию, можно отделить то, что может быть сгруппировано по какому-либо типу.
- Часто изменяемые свойства и методы могут быть сгруппированы в отдельный тип, таким образом основной тип не будет подвергаться изменениям. Это усиливает принцип ООП: «Код должен быть открытым для Расширения, но закрытым для Изменений».

Таким образом, можно сказать, что абстракция упрощает рассмотрение сущности, выделяя существенные свойства на первый план, параллельно и игнорируя второстепенные свойства[24].

Принцип инкапсуляции в ООП

Инкапсуляция - принцип объектно-ориентированного программирования, позволяющий собрать объект в пределах одной структуры или массива, убрав способ обработки данных и сами данные от «чужих глаз». Это одновременно и облегчает конечному пользователю работу с программой, и защищает данные и само приложение от постороннего вмешательства. Пользователь может работать со всем функционалом через интерфейс, не задумываясь над тем, как программа работает. Инкапсуляцию применяют:

- когда нужно сохранить некоторый участок кода без изменений со стороны пользователя;
- когда нужно ограничить доступ к коду - в связи с уникальностью используемых техник, которые автор хочет оставить «при себе»;
- когда изменение кода повлечёт за собой неработоспособность программы или её взлом.

Она неразрывна с абстракцией, и по сути благодаря ей она и работает. Инкапсуляция - это своеобразный клей (или синяя изолента), которым склеивают разные чертежи в один. То есть совмещение деталей для создания своей - это и есть инкапсуляция. Причём при совмещении можно не описывать детали этого совмещения (то есть члены класса могут быть приватными), таким образом помогая абстрагироваться тем, кто этот чертёж использует.

Следует отметить, что инкапсуляция присуща не только объектно-ориентированному программированию. Например, в процедурном программировании инкапсуляция осуществляется, например, при помощи подпрограмм (как вы помните, в ActionScript подпрограммы называются функциями). Однако в объектно-ориентированном программировании ее применение значительно расширяется благодаря введению понятия объекта[25].

Ниже представлены некоторые преимущества инкапсуляции:

- Можно защитить внутреннее состояние объекта с помощью сокрытия его атрибутов.
- Это улучшает модульное построение кода, так как предотвращает взаимодействие объектов неожиданными способами.
- Повышается практичность кода.
- Это поддерживает договорные отношения конкретного объекта.
- Инкапсуляция облегчает поддержку ПО.
- Изменения в коде могут производиться независимо друг от друга.

Наследование классов в ООП

Понятие класса приводит нас к понятию наследования. В повседневной жизни часто сталкиваемся с разбиением классом на подклассы: например, класс животные можно разбить на подклассы млекопитающие, земноводные, насекомые, птицы и т. д. Класс наземный транспорт делится на классы автомобили, грузовики, автобусы, мотоциклы и т. д.

То есть, наследование - это способность в объектно-ориентированном программировании построить новый класс на основе уже заданного. При этом функционал может как полностью совпадать, так и отличаться. Класс-донор называется в таком случае родительским или базовым, а его «потомок» - наследником, дочерним классом. Существует также множественное наследование, при котором у класса-наследника может быть несколько «родителей». При этом класс наследует методы всех своих отцов и матерей, что часто приводит к ошибкам. Наследование требует определения ещё одного понятия: прототип - объект-образец, на основе которого «рождаются» другие объекты, полностью копируя его или изменяясь в процессе. При изменении в прототипе в копиях также происходят соответствующие изменения[26].

Преимущества наследования следующие:

- Усовершенствованное повторное использование кода.
- Устанавливается логическое отношение «is a» (является кем-то, чем-то). Например: Dog is an animal. (Собака является животным).
- Модуляризация кода.
- Исключаются повторения.

Недостаток наследования следующий:

- Сильная связанность: подкласс зависит от реализации родительского класса, что делает код сильно связанным.

Принцип полиморфизма

Полиморфизм - способность объектов самим определять, какие методы они должны применить в зависимости от того, где именно в коде они находятся. То есть, объект может изменяться в зависимости от своего местоположения и действовать по-разному, что позволяет не заводить лишних структур. Иначе говоря: один интерфейс - множество решений. Полиморфизм позволяет повысить

процент повторного использования кода и сократить этим самым размер программы и временные затраты на её написание[27].

Механизмы, посредством которых в языках программирования обеспечивается поддержка полиморфизма, очень разнообразны [28].

Полиморфизм ни в коем случае нельзя рассматривать отдельно от других фундаментальных понятий - абстракция, инкапсуляция и наследование.

Преимущества полиморфизма следующие:

- Создание повторно используемого кода. То есть, как только класс создан, реализован и протестирован, он может свободно использоваться без заботы о том, что конкретно в нем написано.
- Это обеспечивает более универсальный и слабосвязанный код.
- Понижается время компиляции, что ускоряет разработку.
- Динамическое связывание.
- Один и тот же интерфейс может быть использован для создания методов с разными реализациями.
- Вся реализация может быть заменена с помощью использования одинаковых сигнатур метода.

На сегодняшний день самые популярные языки программирования - это объектно-ориентированные, к примеру, C++ и Java. Также существуют языки, которые не предполагают написания программных операторов, а программирование происходит в виде визуального проектирования с помощью интерфейса языка. Примером таких систем являются VisualBasic, Delphiи C++ Builder.

Заключение

В результате написания курсовой работы на тему «Анализ методов современного программирования» было выполнено:

- Изучение основных понятий по данной теме;
- Раскрыть сущность методологии программирования;
- Раскрыта суть модульного программирования;
- Раскрыта сущность структурного программирования;
- Раскрыта сущность объектно-ориентированного программирования;

- Описаны основные понятия, которые связаны с объектно-ориентированным программированием.

В результате проведенного исследования литературы по данной теме можно заключить следующее:

На данный момент существует четыре широко известных в настоящее время методологии программирования – императивного, объектно-ориентированного, логического, функционального.

В результате изучения модульного программирования, можно сказать, что оно является основным способом разработки сложного программного обеспечения.

В результате изучения структурного программирования, можно сказать, оно как правило используется для разработки крупных программных комплексом. А так же следует отметить и то, что программа, которая написана с применением методов структурного программирования понятна, более надежна и облегчает сопровождение.

В результате изучения объектно-ориентированного программирования и его принципов можно сказать, что: абстракция невозможна без инкапсуляции и наследовании, как невозможен полиморфизм без, собственно, наследования. Ну а полиморфизм невозможен ещё и без инкапсуляции, которая банально бесполезна без наследования и полиморфизма.

Библиография

1. Assembler. Учебник для вузов. 2-е изд. — СПб.: Питер, 2010. — 637 с: ил.
2. Акимов П.А., Кайтуков Т.Б., Мозгалева М.Л., Сидоров В.Н. Строительная информатика: учебное пособие. - М.: Издательство АСВ, 2014.-432 стр.
3. Алиев, В. Компьютер — это просто! - учебник. - Спб.: Питер, 2012 г.
4. Васильев А. C#. Объектно-ориентированное программирование: Учебный курс. — СПб.: Питер. 2012. — 320 с.
5. Гурский Д. ActionScript 2.0: программирование во Flash MX 2004. Для профессионалов. —СПб.: Питер. 2004. — 1088 с: ил.
6. Кузнецов, М. В.PHP 5/6 / М. В. Кузнецов, И. В. Симдяпов. — СПб.: БХВ-Петербург, 2010. — 1024 с.

7. Лафоре Р. Объектно-ориентированное программирование в C++. Классика Computer Science. 4-е изд. - СПб.: Питер, 2011. - 928 с: ил.
8. Орлов С. А.Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го поколения. — СПб.: Питер, 2014. — 688 с: ил.
9. Орлов С. А.Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го поколения. — СПб.: Питер, 2014. — 688 с: ил.
10. Орлов С. А.Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го
11. поколения. — СПб.: Питер, 2014. — 688 с: ил.
12. Рапаков Г. Г., Ржеуцкая С. Ю. Программирование на языке Pascal. — СПб.: БХВ-Петербург, 2004. - 480 с.
13. Хабнбуллин И. Ш. Java 7. — СПб.: БХВ-Петербург, 2012. – 738с.
14. Аппаратное обеспечение [онлайн] -URL: http://linchakin.com/%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/%D0% (дата обращения 07.11.2016).
15. КОНСПЕКТ ОБЗОРНОЙ ЛЕКЦИИ по курсу «ЭВМ и программирование» доцента кафедры ИВТ, к.т. н. Ливак Е.Н. на тему Основные технологии разработки программного обеспечения [онлайн] - URL: http://mf.grsu.by/Kafedry/prikl_mat/academic_process/045/Lec_31.doc (дата обращения 03.11.2016)
16. Методология программирования [онлайн] - URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%B (дата обращения 11.11.2016).
17. Модульное программирование [онлайн] - URL: http://helpiks.org/5-51370.html (дата обращения 01.11.2016)
18. Модульное программирование [онлайн] - URL: http://uchebnik-online.net/book/701-informatika-uchebnoe-posobie/11-modulnoe-programmirovanie.html (дата обращения 14.11.2016)
19. Модульное программирование [онлайн] - URL:http://h-l-l.ru/publ/36-1-0-57 (дата обращения 14.11.2016)
20. МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ: ОСНОВНЫЕ ПОНЯТИЯ И НАЗНАЧЕНИЕ // А.Ю. Сикиржицкий [онлайн] - URL:http://media.miu.by/files/store/items/rnpsmu-pipsn/iv/mspp_2013_33.pdf (дата обращения 28.10.2016)
21. Модульное программирование[онлайн] -URL: http://webkonspect.com/?room=profile&id=20978&labelid=210661(дата обращения 16.11.2016).

22. Парадигма структурного программирования Информатика (Практика программирования) [онлайн] - URL: http://foxford.ru/wiki/informatika/paradigma-strukturnogo-programmirovaniya (дата обращения 19.11.2016)
23. Понятие о структурном программировании. Модульный принцип программирования. Подпрограммы. Принципы проектирования программ "сверху-вниз" и "снизу-вверх". Объектно-ориентированное программирование [онлайн] - URL: http://www.altstu.ru/media/f/Tema-20-Strukturnoe-programmirovanie.pdf (дата обращения 14.11.2016))
24. Программирование. Методология программирования. [онлайн] - URL: http://izi.vlsu.ru/teach/books/905/theory.html (дата обращения 11.11.2016).
25. Структурное программирование [онлайн] - URL: http://www.maksakov-sa.ru/TehProgram/StrProgr/index.html (дата обращения 14.11.2016)
26. Типы программирования. Часть 1. Структурное программирование. Циклы [онлайн] - URL: http://www.tryobj.com/56-oprog-17.html (дата обращения 14.11.2016))

1. Алиев, В. Компьютер — это просто! - учебник. - Спб.: Питер, 2012 г. – с.8 ↑

2. Аппаратное обеспечение [онлайн] -URL: http://linchakin.com/%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/%D0% (дата обращения 07.11.2016) ↑

3. Орлов С. А.Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го поколения. — СПб.: Питер, 2014. — с. 25 ↑

4. Акимов П.А., Кайтуков Т.Б., Мозгалева М.Л., Сидоров В.Н. Строительная информатика: учебное пособие. - М.: Издательство АСВ, 2014.- с. 14 ↑

5. Акимов П.А., Кайтуков Т.Б., Мозгалева М.Л., Сидоров В.Н. Строительная информатика: учебное пособие. - М.: Издательство АСВ, 2014.-с. 14 - 15 ↑

6. Кузнецов, М. В.PHP 5/6 / М. В. Кузнецов, И. В. Симдяпов. — СПб.: БХВ-Петербург, 2010. — с. 997 ↑

7. Программирование. Методология программирования. [онлайн] - URL: http://izi.vlsu.ru/teach/books/905/theory.html (дата обращения 11.11.2016). ↑

8. Методология программирования [онлайн] - URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%B (дата обращения 11.11.2016). ↑

9. Методология программирования [онлайн] - URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%B (дата обращения 11.11.2016). ↑

10. Assembler. Учебник для вузов. 2-е изд. — СПб.: Питер, 2010. — с. 325 ↑

11. Модульное программирование[онлайн] -URL: http://webkonspect.com/?room=profile&id=20978&labelid=210661(дата обращения 16.11.2016). ↑

12. Модульное программирование [онлайн] - URL: http://uchebnik-online.net/book/701-informatika-uchebnoe-posobie/11-modulnoe-programmirovanie.html (дата обращения 14.11.2016) ↑

13. Модульное программирование [онлайн] - URL:http://h-l-l.ru/publ/36-1-0-57 (дата обращения 14.11.2016)). ↑

14. МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ: ОСНОВНЫЕ ПОНЯТИЯ И НАЗНАЧЕНИЕ // А.Ю. Сикиржицкий [онлайн] - URL:http://media.miu.by/files/store/items/rnpsmu-pipsn/iv/mspp_2013_33.pdf (дата обращения 28.10.2016) ↑

15. Модульное программирование [онлайн] - URL: http://helpiks.org/5-51370.html (дата обращения 01.11.2016) ↑

16. Assembler. Учебник для вузов. 2-е изд. — СПб.: Питер, 2010. — с. 325 ↑

17. Понятие о структурном программировании. Модульный принцип программирования. Подпрограммы. Принципы проектирования программ "сверху-вниз" и "снизу-вверх". Объектно-ориентированное программирование [онлайн] - URL: http://www.altstu.ru/media/f/Tema-20-Strukturnoe-programmirovanie.pdf (дата обращения 14.11.2016)) ↑

18. Типы программирования. Часть 1. Структурное программирование. Циклы [онлайн] - URL: http://www.tryobj.com/56-oprog-17.html (дата обращения 14.11.2016)) ↑

19. Структурное программирование [онлайн] - URL: http://www.maksakov-sa.ru/TehProgram/StrProgr/index.html (дата обращения 14.11.2016) ↑

20. Парадигма структурного программирования Информатика (Практика программирования) [онлайн] - URL: http://foxford.ru/wiki/informatika/paradigma-strukturnogo-programmirovaniya (дата обращения 19.11.2016) ↑

21. Васильев А. C#. Объектно-ориентированное программирование: Учебный курс. — СПб.: Питер. 2012. —с. 35 - 36 ↑

22. Хабнбуллин И. Ш. Java 7. — СПб.: БХВ-Петербург, 2012. – с. 76 ↑

23. Рапаков Г. Г., Ржеуцкая С. Ю. Программирование на языке Pascal. — СПб.: БХВ-Петербург,2004. – с. 375 ↑

24. Орлов С. А.Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го

    поколения. — СПб.: Питер, 2014. — с. 400 ↑

25. Гурский Д. ActionScript 2.0: программирование во Flash MX 2004. Для профессионалов. —

    СПб.: Питер. 2004. — с. 366 ↑

26. Лафоре Р. Объектно-ориентированное программирование в C++. Классика Computer Science. 4-е изд. - СПб.: Питер, 2011. – с. 40-41 ↑

27. КОНСПЕКТ ОБЗОРНОЙ ЛЕКЦИИ по курсу «ЭВМ и программирование» доцента кафедры ИВТ, к.т. н. Ливак Е.Н. на тему Основные технологии разработки программного обеспечения [онлайн] - URL: http://mf.grsu.by/Kafedry/prikl_mat/academic_process/045/Lec_31.doc (дата обращения 03.11.2016) ↑

28. Орлов С. А.Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го

поколения. — СПб.: Питер, 2014. — с. 357 ↑