**Концепции современного программирования. Парадигма модульного программирования. Парадигма программирования на машинном языке Парадигма языков высокого уровня. Парадигма процедурного программирования. Парадигма структурного программирования. Парадигма модульного программирования Парадигма объектно-ориентированного программирования. Функциональное программирование.**

***Это вопросы из программы для экзамена.***

Изначально программирование возникло как математическая дисциплина и первым применением новой области знания стало решение естественно-научных задач. Не будет большой ошибкой сказать, что первые программируемые вычислительные машины возникли в ходе создания ядерного оружия в США и СССР. При этом первыми программистами были математики, физики, электронщики. На первоначальном этапе программирование было искусством, развлечением, но никак не ремеслом. Основными проблемами, стоявшими перед программистами были мизерные (по современным меркам) объемы оперативной памяти и низкое быстродействие компьютеров того времени. Программы писались на машинных языках в двоичном представлении. При написании программ на машинных языках часто допускались ошибки, причем из-за невозможности их структурировать сопровождение кода было практически невозможным. Кроме того, программа в машинных кодах была сложна для понимания (**парадигма программирования на машинном языке**). Такое программирование начиналось без каких-либо единых подходов, поскольку программы писались в машинных кодах и программирование сильно зависело от конкретной машины. Однако с развитием вычислительной техники и с построением серийных компьютеров возникли универсальные языки программирования высокого уровня. Первым известным языком программирования был язык Фортран. Этот язык, название которого является сокращением от "транслятор формул", представлял собой уже в определенном смысле новую парадигму программирования, которую называют **парадигмой языков высокого уровня**. Программирование на языке Фортран представляет собой написание программ с использованием формализованных команд, которые в определенном смысле можно считать математическим текстом. С языка Фортран началось алгоритмическое программирование, когда появился более-менее стандартный язык для написания программ. При этом стало возможным писать программы" впрок", когда при создании программ можно было использовать ранее созданные тексты программ. Многие часто используемые подпрограммы стали оформляться в виде специальных библиотек. С дальнейшим развитием компьютерной техники и расширением сферы программирования возникла следующая парадигма программирования, которую называют **"парадигма процедурного (императивного)программирования"**.

Процедурные языки дают программисту возможность разбить программу обработки информации на несколько процедур более низкого уровня. Такие процедуры более низкого уровня определяют структуру программы в целом. Последовательные обращения к этим процедурам управляют выполнением программы, состоящей из процедур. Программа прекращает работу тогда, когда исчерпывается список вызовов процедур. Процедурное программирование позволяет использовать написанные ранее подпрограммы, оформленные в виде подключаемых библиотек. Именно с этого момента стало возможным создания сложных программных комплексов. Если первоначально программы писались одним программистом в каждый момент времени, то при создании сложных программных комплексов необходимо привлекать целый коллектив программистов, которые работали одновременно. Это потребовало изменений в организацию процесса программирования. Процедурное программирование позволяло разделить процесс программирования на отдельные задачи, которые можно было выполнять одновременно разными программистами. При этом вместо написания большой программы, создавались относительно небольшие подпрограммы, что позволяло значительно легче отлаживать программы и модернизировать ранее созданные комплексы.

Эта новая парадигма программирования была прогрессивной по сравнению с парадигмой программирования на машинном языке, в нее было добавлено главное средство структурирования — процедуры. Более мелкие функции не только проще понять, но также проще отладить. Но, с другой стороны, процедурное программирование ограничивает повторное использование кода. Притом очень часто программисты писали "макаронные" программы — программы, выполнение которых напоминало распутывание макарон в тарелке спагетти. И, в конце концов, стало понятно, что концентрация внимания на данных при разработке программ методами процедурного программирования сама по себе вызывает проблемы. Так как данные и процедура разделены, нет инкапсуляции данных. Это приводит к тому, что каждая процедура должна знать, что делать с данными. К сожалению, когда процедура ведет себя плохо, она может испортить данные, если выполнит неправильные действия над ними. Поскольку каждая процедура должна была дублировать информацию о доступе к данным, изменение представления данных приводило к изменениям всех тех мест программы, в которых этот доступ осуществлялся. Таким образом, даже маленькая поправка могла привести к целому ряду изменений во всей программе — другими словами к кошмару при сопровождении системы программного обеспечения.

Если на начальном этапе развития программирования схема создания программы была следующая:

1. Построение алгоритма и его формальное описание
2. Программирование (кодирование алгоритма)
3. Отладка программы
4. Использование программы

то с развитием процедурного подхода появился еще один важный этап- *проектирование*. С появлением этапа проектирования программы становятся программными комплексами. Развитое процедурное программирование становится новой парадигмой программирования, которая называется **парадигмой структурного программирования**.

Под структурным программированием понимают такой метод разработки программ, который начинается с тщательного проектирования структуры программы методом "сверху вниз". При этом каждый блок должен быть спроектирован таким образом, чтобы его создание и отладка могла быть автономной. Часто под структурным программированием понимают и использование "правил хорошего тона", например исключение оператором GOTO, а также подробное документирование текстов программ. Идеология структурного программирование имеет своих основных авторов, к которым относятся такие известные ученые, как Э.В.Дейкстра и Н.Вирт.

В модульном программировании (язык Modula2), была сделана попытка устранить некоторые недостатки, найденные в процедурном и структурном программировании. Модульное программирование разбивает программу на ряд составляющих компонентов, или модулей. В отличие от процедурного программирования, которое разделяет данные и процедуры, модули объединяют их (инкапсулируют). Модуль состоит из самих данных и процедур для обработки данных. Когда другим частям программы нужно использовать модуль, они просто обращаются к интерфейсу модуля. Модули скрывают всю внутреннюю информацию от остальных частей программы. Основываясь на этом, довольно просто объяснить, что такое состояние: модуль сохраняет информацию о состоянии, причем эта информация может измениться (**парадигма модульного программирования**).

Однако модульное программирование имеет свои недостатки. Модули не расширяемы, это означает невозможность производить пошаговые изменения модуля без непосредственного доступа к коду и его прямого изменения. Кроме того, при разработке одного модуля нельзя использовать другой, иначе как через передачу (делегирование) функций. И хотя в модуле можно определить тип, один модуль не может использовать тип, определенный в другом модуле. В модульных и процедурных языках у структурированных и неструктурированных данных есть свой "тип". Тип легче всего определить как формат данных в оперативной памяти.

Следующая парадигма программирования - **"парадигма объектно-ориентированного программирования".** Конечно, слово "следующая" нужно понимать не в хронологическом порядке, а по распространению. Поскольку, если структурное программирование было создано в 70-х годах, то объектно-ориентированное программирование появилось в языке Simula, в 1967 году. Объектно-ориентированное программирование явилось новым шагом в технологии программирования и позволило создавать сложные программные комплексы. В настоящее время подход объектно-ориентированного программирования является стандартом для промышленного программирования. Для реализации объектно-ориентированного подхода необходима соответствующая поддержка со стороны языковых конструкций. К числу наиболее распространенных объектно-ориентированных языков относятся:

* C++
* Java
* C#
* Object Pascal
* VB.NET
* Perl
* PHP
* JavaScript
* и многие другие.

Объектно-ориентированный подход оказал большое влияние на архитектуру операционной системы Microsoft Windows. Хотя в начале основным средством для программирования под Windows был язык C, понятие "окна" в Windows вполне соответствовало понятию объекта или класса.

Любая программа - это обработка данных, однако часто программы создаются для обработки не просто данных, а данных описывающих те или иные объекты (персоны, машины, математические функции и т.д.).Понятие "объекта" включает в себя некоторую информацию, описывающую состояние объекта (при этом эта информация может быть видимой извне объекта, а может быть и невидимой изнутри объекта).Далее объект может уметь выполнять определенные команды, которые поступают из внешней среды по строго установленному интерфейсу. В объектно-ориентированном программировании это называется **инкапсуляцией**. Еще процедурное программирование учит, что необходимо минимизировать повторяющиеся куски программы. Необходимо писать такие программы, которые будут удовлетворять требованиям повторного использования кода. В объектно-ориентированном программировании для этого используются механизмы **наследования** и **полиморфизма**.

Хотя объектов может быть довольно много, но часто рассматриваемые объекты имеют общие черты. Для того, чтобы не создавать повторяющегося кода при описании объектов, был придуман механизм наследования. Пусть моделируется транспортная сеть, которая состоит из автобусов, маршрутных такси, трамваев и троллейбусов. Соответственно, должно быть четыре объекта: автобус, такси, трамвай и автобус. Каждый объект характеризуется рядом показателей: положение, скорость, количество пассажиров, возможность проехать по выбранному маршруту и т.д. Ясно, что некоторые характеристики присущи всем четырем объектам, далее часть характеристик (возможность проехать по данной улице) присущи только трамваю и троллейбусу потому, что для этого транспорта необходимо заранее проложить провода, а для трамвая и рельсы. Такие объекты можно реализовать на основе объектно-ориентированного подхода. Первым делом вводится объект "транспортное средство", все другие объекты будут наследниками от этого объекта. Далее вводятся объекты: "автотранспорт" и "электротранспорт". Эти объекты будут наследниками от объекта "транспортное средство". Наконец, вводятся объекты: "автобус", "такси", "трамвай" и "троллейбус". При этом первые два объекта наследники от объекта "автотранспорт", а последние два от объекта "электротранспорт". При наследовании объекта будет доступен код, который создавался для родительского класса. Технология полиморфизма позволяет, грубо говоря, модифицировать созданный ранее код для наследуемого класса.

**Создать UML-диаграмму классов, иллюстрирующую транспортную сеть. Дата сдачи рисунка с диаграммами 6 марта.**

В последнее время набирает популярность функциональное программирование. Функциональное программирование довольно радикально отличается от рассмотренных парадигм программирования. Если рассмотренные парадигмы программирования были основаны на императивном программировании, то есть по шаговым заданием операций, то функциональное программирование основано на вычислениях функций. В программах, написанных с помощью функционального программирования, нет состояния программы в каждый момент времени. Считается, что функциональное программирование намного ближе к математическому мышлению. Действительно, функциональное программирование основано на \lambda-исчислении, известном в математической логики, посвященной теории вычислимости. Языки функционального программирования:

* LISP
* Haskell
* ML
* F#

Язык LISP был разработан в 1958 году, что раньше таких известных процедурных языков, как ALGOL-60, Pascal, C...Однако именно сейчас увеличился интерес к функциональным языкам программирования, что подтверждается поддержкой языка F# в Microsoft Visual Studio 2010. Функциональное программирование позволяет больше думать и меньше писать.

**Ключевые термины**

**Парадигма объектно-ориентированного программирования**- использование объектов и классов, а также технологий инкапсуляции, наследования и полиморфизма.

**Парадигма процедурного программирования**- систематическое использование отдельных небольших подпрограмм или процедур.

**Парадигма структурного программирования**- метод разработки программ "сверху вниз", использование специальных правил хорошего тона при программировании.

**Парадигма языков высокого уровня**- использование машинно-независимых языков программирования, запись программ с помощью операторов языка программирования а не машинных команд.