**Тема №1. Введение. Знакомство с содержанием дисциплины. Обзор современных технологий разработки программного обеспечения**

Цель: рассмотреть этапы развития программирования как науки, повторить основные понятия и определения, классификцию программного обеспечения.

План:

1.Технология программирования в историческом аспекте

2. Основные понятия и определения

3. Классификация программного обеспечения

**1 Технология программирования в историческом аспекте**

Чтобы разобраться в существующих технологиях программирования и определить основные тенденции их развития, целесообразно рассматривать эти технологии в историческом контексте, выделяя основные этапы развития программирования как науки.

**Первый этап** — «стихийное» программирование. Этот этап охватывает период от момента появления первых вычислительных машин до середины 60-х годов XX в. В этот период практически отсутствовали сформулированные технологии и программирование фактически было искусством. Первые программы имели простейшую структуру. Они состояли из собственно программы на машинном языке и обрабатываемых ею данных.

Сложность программ в машинных кодах ограничивалась способностью программиста одновременно мысленно, отслеживать последовательность выполняемых операций и местонахождение данных при программировании.

Появление ассемблеров позволило вместо двоичных или 16-ричных кодов использовать символические имена данных и мнемоники кодов операций. В результате программы стали более «читаемыми».

Создание языков программирования высокого уровня, таких как FORTRAN и ALGOL, существенно упростило программирование вычислений, снизив уровень детализации операций. Это, в свою очередь, позволило увеличить сложность программ.

Революционным было появление в языках средств, позволяющих оперировать подпрограммами. Подпрограммы можно было сохранять и использовать в других программах. В результате были созданы огромные библиотеки расчетных и служебных подпрограмм, которые по мере надобности вызывались из разрабатываемой программы.

Типичная программа того времени состояла из основной программы, области глобальных данных и набора подпрограмм (в основном библиотечных), выполняющих обработку всех данных или их части. Слабым местом такой архитектуры было то, что при увеличении количества подпрограмм возрастала вероятность искажения части глобальных данных какой-либо подпрограммой.

Например, подпрограмма поиска корней уравнения на заданном интервале по методу деления отрезка пополам меняет величину интервала. Если при выходе из подпрограммы не предусмотреть восстановления первоначального интервала, то в глобальной области окажется неверное значение интервала. Чтобы сократить количество таких ошибок в подпрограммах, было предложено размещать локальные данные.

Сложность разрабатываемого программного обеспечения при использовании подпрограмм с локальными данными по-прежнему ограничивалась возможностью программиста отслеживать процессы обработки данных, но уже на новом уровне. Однако появление средств поддержки подпрограмм позволило осуществлять разработку программного обеспечения нескольким программистам параллельно.

В начале 60-х годов XX в. разразился «кризис программирования». Он выражался в том, что фирмы, взявшиеся за разработку сложного программного обеспечения, такого как операционные системы, срывали все сроки завершения проектов. Проект устаревал раньше, чем был готов к внедрению, увеличивалась его стоимость, и в результате многие проекты так никогда и не были завершены.

Объективно все это было вызвано несовершенством технологии программирования. Прежде всего стихийно использовалась разработка «снизу—вверх» — подход, при котором вначале проектировали и реализовывали сравнительно простые подпрограммы, из которых затем пытались построить сложную программу. В отсутствие четких моделей описания подпрограмм и методов их проектирования создание каждой подпрограммы превращалось в непростую задачу, интерфейсы подпрограмм получались сложными, и при сборке программного продукта выявлялось большое количество ошибок согласования. Исправление таких ошибок, как правило, требовало серьезного изменения уже разработанных подпрограмм, что еще более осложняло ситуацию, так как при этом в программу часто вносились новые ошибки, которые также необходимо было исправлять. В конечном итоге процесс тестирования и отладки программ занимал более 80% времени разработки, если вообще когда-нибудь заканчивался. На повестке дня самым серьезным образом стоял вопрос разработки технологии создания сложных программных продуктов, снижающей вероятность ошибок проектирования.

Анализ причин возникновения большинства ошибок позволил сформулировать новый подход к программированию, который был назван «структурным».

**Второй этап** — структурный подход к программированию (60—70-е годы XX в.). Структурный подход к программированию представляет собой совокупность рекомендуемых технологических приемов, охватывающих выполнение всех этапов разработки программного обеспечения. В основе структурного подхода лежит декомпозиция (разбиение на части) сложных систем с целью последующей реализации в виде отдельных небольших (до 40—50 операторов) подпрограмм. С появлением других принципов декомпозиции (объектного, логического и т. д.) данный способ получил название процедурной декомпозиции.

В отличие от используемого ранее процедурного подхода к декомпозиции, структурный подход требовал представления задачи в виде иерархии подзадач простейшей структуры.

Проектирование, таким образом, осуществлялось «сверху—вниз» и подразумевало реализацию общей идеи, обеспечивая проработку интерфейсов подпрограмм. Одновременно вводились ограничения на конструкции алгоритмов, рекомендовались формальные модели их описания, а также специальный метод проектирования алгоритмов — метод пошаговой детализации.

Поддержка принципов структурного программирования была заложена в основу так называемых процедурных языков программирования. Как правило, они включали основные «структурные» операторы передачи управления, поддерживали вложение подпрограмм, локализацию и ограничение области «видимости» данных. Среди наиболее известных языков этой группы стоит назвать PL/1, ALGOL-68, Pascal, С. Одновременно со структурным программированием появилось огромное количество языков, базирующихся на других концепциях, но большинство из них не выдержало конкуренции.

Какие-то языки были просто забыты, идеи других были в дальнейшем использованы в следующих версиях развиваемых языков. Дальнейший рост сложности и размеров разрабатываемого программного обеспечения потребовал развития структурирования данных. Как следствие этого в языках появляется возможность определения пользовательских типов данных. Одновременно усилилось стремление разграничить доступ к глобальным данным программы, чтобы уменьшить количество ошибок, возникающих при работе с глобальными данными. В результате появилась и начала развиваться технология модульного программирования.

Модульное программирование предполагает выделение групп подпрограмм, использующих одни и те же глобальные данные в отдельно компилируемые модули (библиотеки подпрограмм), например модуль графических ресурсов, модуль подпрограмм вывода на принтер (рис. 1.4). Связи между модулями при использовании данной технологии осуществляются через специальный интерфейс, в то время как доступ к реализации модуля (телам подпрограмм и некоторым «внутренним» переменным) запрещен. Эту технологию поддерживают современные версии языков Pascal и С (C++), языки Ада и Modula.

Использование модульного программирования существенно упростило разработку программного обеспечения несколькими программистами. Теперь каждый из них мог разрабатывать свои модули независимо, обеспечивая взаимодействие модулей через специально оговоренные межмодульные интерфейсы. Кроме того, модули в дальнейшем без изменений можно было использовать в других разработках, что повысило производительность труда программистов.

Практика показала, что структурный подход в сочетании с модульным программированием позволяет получать достаточно надежные программы, размер которых не превышает 100 000 операторов. Узким местом модульного программирования является то, что ошибка в интерфейсе при вызове подпрограммы выявляется только при выполнении программы (из-за раздельной компиляции модулей обнаружить эти ошибки раньше невозможно). При увеличении размера программы обычно возрастает сложность межмодульных интерфейсов, и с некоторого момента предусмотреть взаимовлияние отдельных частей программы становится практически невозможно. Для разработки программного обеспечения большого объема было предложено использовать объектный подход.

**Третий этап** — объектный подход к программированию (с середины 80-х до конца 90-х годов XX в.). Объектно-ориентированное программирование определяется как технология создания сложного программного обеспечения, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного типа {класса), а классы образуют иерархию с наследованием свойств [10, 24, 29]. Взаимодействие программных объектов в такой системе осуществляется путем передачи сообщений (рис. 1.5). Объектная структура программы впервые была использована в языке имитационного моделирования сложных систем Simula, появившемся еще в 60-х годах XX в. Естественный для языков моделирования способ представления программы получил развитие в другом специализированном языке моделирования языке Smalltalk (70-е годы XX в.), а затем был использован в новых версиях универсальных языков программирования, таких как Pascal, C++, Modula, Java.

Основным достоинством объектно-ориентированного программирования по сравнению с модульным программированием является «более естественная» декомпозиция программного обеспечения, которая существенно облегчает его разработку. Это приводит к более полной локализации данных и интегрированию их с подпрограммами обработки, что позволяет вести практически независимую разработку отдельных частей (объектов) программы. Кроме этого, объектный подход предлагает новые способы организации программ, основанные на механизмах наследования, полиморфизма, композиции, наполнения. Эти механизмы позволяют конструировать сложные объекты из сравнительно простых. В результате существенно увеличивается показатель повторного использования кодов и появляется возможность создания библиотек классов для различных применений.

Бурное развитие технологий программирования, основанных на объектном подходе, позволило решить многие проблемы. Так, были созданы среды, поддерживающие визуальное программирование, например Delphi, C++ Builder, Visual C++ и т. д. При использовании визуальной среды у программиста появляется возможность проектировать некоторую часть, например интерфейсы будущего продукта, с применением визуальных средств добавления и настройки специальных библиотечных компонентов. Результатом визуального проектирования является заготовка будущей программы, в которую уже внесены соответствующие коды.

Использование объектного подхода имеет много преимуществ, однако его конкретная реализация в объектно-ориентированных языках программирования, таких как Pascal и C++, имеет существенные недостатки:

1) фактически отсутствуют стандарты компоновки двоичных результатов компиляции объектов в единое целое даже в пределах одного языка программирования. Компоновка объектов, полученных разными компиляторами C++, в лучшем случае проблематична, что приводит к необходимости разработки программного обеспечения с использованием средств и возможностей одного языка программирования высокого уровня и одного компилятора, а значит, требует наличия исходных кодов используемых библиотек классов;

2) изменение реализации одного из программных объектов, как минимум, связано с перекомпиляцией соответствующего модуля и перекомпоновкой всего программного обеспечения, использующего данный объект.

Таким образом, при использовании этих языков программирования сохраняется зависимость модулей программного обеспечения от адресов экспортируемых полей и методов, а также структур и форматов данных. Эта зависимость объективна, так как модули должны взаимодействовать между собой, обращаясь к ресурсам друг друга. Связи модулей нельзя разорвать, но можно попробовать стандартизировать их взаимодействие, на чем и основан компонентный подход к программированию.

**Четвертый этап** — компонентный подход и CASE-технологии (с середины 90-х годов XX в. до нашего времени). Компонентный подход предполагает построение программного обеспечения из отдельных компонентов физически отдельно существующих частей программного обеспечения, которые взаимодействуют между собой через стандартизованные двоичные интерфейсы. В отличие от обычных объектов объекты-компоненты можно собрать в динамически вызываемые библиотеки или исполняемые файлы, распространять в двоичном виде (без исходных текстов) и использовать в любом языке программирования, поддерживающем соответствующую технологию. На сегодня рынок объектов стал реальностью, так, в Интернете существуют узлы, предоставляющие большое количество компонентов, рекламой компонентов забиты журналы. Это позволяет программистам создавать продукты, хотя бы частично состоящие из повторно использованных частей, т. е. использовать технологию, хорошо зарекомендовавшую себя в области проектирования аппаратуры.

Компонентный подход лежит в основе технологий, разработанных на базе COM (Component Object Model — компонентная модель объектов), и технологии создания распределенных приложений CORBA (Common Object Request Broker Architecture — общая архитектура с посредником обработки запросов объектов). Эти технологии используют сходные принципы и различаются лишь особенностями их реализации.

Технология СОМ фирмы Microsoft является развитием технологии OLE I (Object Linking and Embedding — связывание и внедрение объектов), которая использовалась в ранних версиях Windows для создания составных документов. Технология СОМ определяет общую парадигму взаимодействия программ любых типов: библиотек, приложений, операционной системы, т. е. позволяет одной части программного обеспечения использовать функции (службы), предоставляемые другой, независимо от того, функционируют ли эти части в пределах одного процесса, в разных процессах на одном компьютере или на разных компьютерах (рис. 1.6). Модификация СОМ, обеспечивающая передачу вызовов между компьютерами, называется DCOM (Distributed COM — распределенная СОМ).

По технологии СОМ приложение предоставляет свои службы, используя специальные объекты — объекты СОМ, которые являются экземплярами классов СОМ. Объект СОМ так же, как обычный объект, включает поля и методы, но в отличие от обычных объектов каждый объект СОМ может реализовывать несколько интерфейсов, обеспечивающих доступ к его полям и функциям. Это достигается за счет организации отдельной таблицы адресов методов для каждого интерфейса (по типу таблиц виртуальных методов). При этом интерфейс обычно объединяет несколько однотипных функций. Кроме того, классы СОМ поддерживают наследование интерфейсов, но не поддерживают наследования реализации, т. е. не наследуют код методов, хотя при необходимости объект класса-потомка может вызвать метод родителя.

На базе технологии СОМ и ее распределенной версии DCOM были разработаны компонентные технологии, решающие различные задачи разработки программного обеспечения. OLE-automation или просто Automation (автоматизация) — технология создания программируемых приложений, обеспечивающая программируемый доступ к внутренним службам этих приложений. Вводит понятие диспинтерфейса (dispinterface) — специального интерфейса, облегчающего вызов функций объекта. Эту технологию поддерживает, например, Microsoft Excel, предоставляя другим приложениям свои службы.

ActiveX — технология, построенная на базе OLE-automation, предназначена для создания программного обеспечения как сосредоточенного на одном компьютере, так и распределенного в сети. Предполагает использование визуального программирования для создания компонентов — элементов управления ActiveX. Полученные таким образом элементы управления можно устанавливать на компьютер дистанционно с удаленного сервера, причем устанавливаемый код зависит от используемой операционной системы. Это позволяет применять элементы управления ActiveX в клиентских частях приложений Интернета.

Технология CORBA, разработанная группой компаний ОМС (Object Management Group — группа внедрения объектной технологии программирования), реализует подход, аналогичный СОМ, на базе объектов и интерфейсов CORBA. Программное ядро CORBA реализовано для всех основных аппаратных и программных платформ, и потому эту технологию можно использовать для создания распределенного программного обеспечения в гетерогенной (разнородной) вычислительной среде.

Организация взаимодействия между объектами клиента и сервера в CORBA осуществляется с помощью специального посредника, названного VisiBroker, и другого специализированного программного обеспечения.

Отличительной особенностью современного этапа развития технологии программирования, кроме изменения подхода, является создание и внедрение автоматизированных технологий разработки и сопровождения программного обеспечения, которые были названы CASE-технологиями (Computer-Aided Software/ System Engineering — разработка программного обеспечения программных систем с использованием компьютерной поддержки).

Без средств автоматизации разработка достаточно сложного программного обеспечения на настоящий момент становится трудно осуществимой: память человека уже не в состоянии фиксировать все детали, которые необходимо учитывать при разработке программного обеспечения. На сегодня существуют CASE-технологии, поддерживающие как структурный, так и объектный (в том числе и компонентный) подходы к программированию.

Появление нового подхода не означает, что отныне все программное обеспечение будет создаваться из программных компонентов, но анализ существующих проблем разработки сложного программного обеспечения показывает, что он будет применяться достаточно широко.

**2 Основные понятия и определения**

*Технология программирования* — совокупность методов и средств, применяемых в процессе разработки программного обеспечения.

*Программа (program, routine)* — упорядоченная последовательность команд (инструкций) компьютера для решения задачи.

*Программное обеспечение (software)* — совокупность программ обработки данных и необходимых для их эксплуатации документов.

*Задача (problem, task)* — проблема, подлежащая решению.

*Приложение (application)* — программная реализация на компьютере решения 1 задачи.

Термин «задача» в программировании означает единицу работы вычислительной системы, требующую выделения вычислительных ресурсов (процессорного времени, памяти).

Процесс создания программ можно представить как последовательность следующих действий:

1. постановка задачи;
2. алгоритмизация решения задачи;
3. программирование.

*Постановка задачи (problem definition)* — это точная формулировка решения задачи на компьютере с описанием входной и выходной информации.

*Алгоритм* — система точно сформулированных правил, определяющая процесс преобразования допустимых исходных данных (выходной информации) в желаемый результат (выходную информацию) за конечное число шагов.

*Программирование (programming)* — теоретическая и практическая деятельность, связанная с созданием программ.

По отношению к программному обеспечению компьютерные пользователи делятся на следующие группы:

1. системные программисты. Занимаются разработкой, эксплуатацией и сопровождением системного программного обеспечения;
2. прикладные программисты. Осуществляют разработку и отладку программ для решения различных прикладных задач;
3. конечные пользователи. Имеют элементарные навыки работы с компьютером и используемыми ими прикладными программами;
4. администраторы сети. Отвечают за работу вычислительных сетей;
5. администраторы баз данных. Обеспечивают организационную поддержку базы данных.

*Сопровождение программы* — поддержка работоспособности программы, переход на ее новые версии, внесения изменений, исправление ошибок и т. д.

**Основные характеристики программ**:

1. алгоритмическая сложность;
2. состав функций обработки информации;
3. объем файлов, используемых программой;
4. требования к операционной системе (ОС) и техническим средствам обработки, в том числе объем дисковой памяти, размер оперативной памяти для запуска программы, тип процессора, версия ОС, наличие вычислительной сети и т. д.

**Показатели качества программы**:

1. мобильность (многоплатформенность) — независимость от технического комплекса системы обработки данных, ОС, сетевых возможностей, специфики предметной области задачи и т. д.;
2. надежность — устойчивость, точность выполнения предписанных функций обработки, возможность диагностики возникающих ошибок в работе программы;
3. эффективность как с точки зрения требований пользователя, так и расхода вычислительных ресурсов;
4. учет человеческого фактора — дружественный интерфейс, контекстно-зависимая подсказка, хорошая документация;
5. модифицируемость — способность к внесению изменений, например, расширение функций обработки, переход на другую техническую базу обработки и т. п.
6. коммуникативность — максимально возможная интеграция с другими программами, обеспечение обмена данными между программами.

Все программы по характеру использования и категориям пользователей можно разделить на два класса — утилитарные программы и программные продукты.

Утилитарные программы («программы для себя») предназначены для удовлетворения нужд их разработчиков. Чаще всего такие программы выполняют роль отладочных приложений, являются программами решения задач, не предназначенных для широкого распространения.

Программные продукты (изделия) используются для удовлетворения потребностей пользователей, широкого распространения и продажи.

В настоящее время существуют и другие варианты легального распространения программных продуктов, которые появились с использованием глобальных телекоммуникаций:

* freeware — бесплатные программы, свободно распространяемые, поддерживаются самим пользователем, который правомочен вносить в них необходимые изменения;
* shareware некоммерческие (условно-бесплатные) программы, которые могут использоваться, как правило, бесплатно.

Ряд производителей использует OEM-программы (Original Equipment Manufacturer), т. е. встроенные программы, устанавливаемые на компьютеры или поставляемые вместе с компьютерами.

Программный продукт должен быть соответствующим образом подготовлен к эксплуатации (отлажен), иметь необходимую техническую документацию, предоставлять сервис и гарантию надежной работы программы, иметь товарный знак изготовителя, а также наличие кода государственной регистрации.

**3 Классификация программного обеспечения**

Можно выделить три класса программного обеспечения:

1. системное;
2. пакеты прикладных программ (прикладное программное обеспечение);
3. инструментарий технологии программирования (инструментальные средства для разработки программного обеспечения).

Системное программное обеспечение направлено:

* на создание операционной среды функционирования других программ;
* обеспечение надежной и эффективной работы самого компьютера и вычислительной сети;
* проведение диагностики и профилактики аппаратуры компьютера и вычислительных сетей;
* выполнение вспомогательных технологических процессов (копирование, архивация, восстановление файлов программ и БД и т. п.).

*Системное программное обеспечение (System Software)* — совокупность программ и программных комплексов для обеспечения работы компьютера и вычислительных сетей.

Прикладное программное обеспечение служит программным инструментарием решения функциональных задач и является самым многочисленным классом ПО. В данный класс входят программные продукты, выполняющие обработку информации различных предметных областей. Таким образом, *прикладное программное обеспечение* — комплекс взаимосвязанных программ для решения задач определенного класса предметной области.

Инструментарий технологии программирования обеспечивает процесс разработки программ и включает специализированное программное обеспечение, которое является инструментальным средством разработки. Программное обеспечение данного класса поддерживает все технологические этапы процесса проектирования, программирования, отладки и тестирования создаваемых программ. Пользователями данного программноего обеспечения являются системные и прикладные программисты.

**Системное программное обеспечение**

В составе системного программного обеспечения можно выделить две составляющие:

1. базовое программное обеспечение;
2. сервисное программное обеспечение.

*Базовое программное обеспечение* — минимальный набор программных средств, обеспечивающих работу компьютера.

*Сервисное программное обеспечение* — программы и программные комплексы, которые расширяют возможности базового программного обеспечения и организуют более удобную среду работы пользователя.

В базовое программное обеспечение входят:

* операционная система;
* операционные оболочки (текстовые, графические);
* сетевая операционная система.

*Операционная система* — совокупность программных средств, обеспечивающих управление аппаратной частью компьютера и прикладными программами, а также их взаимодействием между собой и пользователем.

Операционная система образует автономную среду, не связанную ни с одним из языков программирования. Любая прикладная программа связана с ОС и может эксплуатироваться только на компьютерах, где имеется аналогичная системная среда. Программа, созданная в среде одной ОС, не функционирует в среде другой ОС, если в ней не обеспечена возможность конвертации (преобразования) программ.

Для работы с операционной системой необходимо овладеть языком этой среды — совокупностью команд, структура которых определяется синтаксисом этого языка.

Операционная система выполняет следующие функции:

* управление работой каждого блока программного обеспечения и их взаимодействием;
* управление выполнением программ;
* организацию хранения информации во внешней памяти;
* взаимодействие пользователя с компьютером (поддержку интерфейса пользователя).

Операционные системы для ПК делятся:

* на одно- и многозадачные (в зависимости от числа параллельного выполнения прикладных процессов);
* одно- и многопользовательские (в зависимости от числа пользователей, одновременно работающих с ОС);
* непереносимые и переносимые на другие типы компьютеров;
* несетевые и сетевые, обеспечивающие работу в ВС ЭВМ.

Операционные системы, как правило, сравнивают по следующим критериям:

* управлению памятью (максимальный объем адресуемого пространства, технические показатели использования памяти);
* функциональным возможностям вспомогательных программ (утилит) в составе ОС;
* наличию компрессии магнитных дисков;
* возможности архивирования файлов;
* поддержке многозадачного режима работы;
* наличию качественной документации;
* условию и сложности процесса инсталляции.

Наиболее распространенными ОС для персональных компьютеров в настоящее время являются:

* Windows 2000 (Microsoft);
* Windows XP (Microsoft);
* UNIX;
* OS/2 (IBM).

Операционная система OS/2 разработана IBM для ПК на основе системной прикладной архитектуры, ранее используемой для больших ЭВМ. Это многозадачная, многопользовательская ОС, обеспечивающая как текстовый, так и графический интерфейс пользователя. OS/2 обеспечивает:

* поддержку графического интерфейса пользователя;
* одновременную обработку нескольких приложений;
* многопоточную обработку нескольких задач одного приложения;
* 32-разрядную обработку данных;
* сжатие данных при записи на магнитный диск;
* защиту памяти.

*Операционные оболочки* — специальные программы, предназначенные для облегчения общения пользователя с командами операционной системы. Операционные оболочки имеют текстовый и графический варианты интерфейса конечного пользователя. Наиболее популярны такие текстовые оболочки MS DOS, как Norton Commander, DOS Navigator, Volkov Commander и др. Эти программы существенно упрощают задание управляющей информации для выполнения команд ОС, уменьшают сложность работы конечного пользователя.

Среди графических оболочек MS DOS можно выделить Windows 3.1, Windows 3.11 for Work Group, которые расширяют набор основных (диспетчер файлов, графический редактор, текстовый редактор и т. д.) и сервисных функций, обеспечивающих создание пользователю систем обработки информации в составе одноранговых локальных сетей.

Расширением базового ПО компьютера является набор сервисных, дополнительно устанавливаемых программ, которые можно классифицировать по функциональному признаку следующим образом:

* программы диагностики работоспособности компьютера;
* антивирусные программы, обеспечивающие защиту компьютера, обнаружение и восстановление зараженных файлов;
* программы обслуживания дисков, обеспечивающие проверку качества поверхности магнитного диска, контроль сохранности файловой системы на логическом и физическом уровнях, сжатие дисков, резервное копирование данных на внешние носители и др.;
* программы архивирования данных, которые обеспечивают процесс сжатия информации с целью уменьшения объема памяти для ее хранения;
* программы обслуживания сети.

Эти программы часто называют утилитами.

*Утилиты* — программы, служащие для выполнения вспомогательных операций обработки данных или обслуживания компьютеров.

**Инструментарий технологии програм**

*Инструментарий технологии программирования* — это программные продукты, предназначенные для поддержки технологии программирования (рисунок 1).

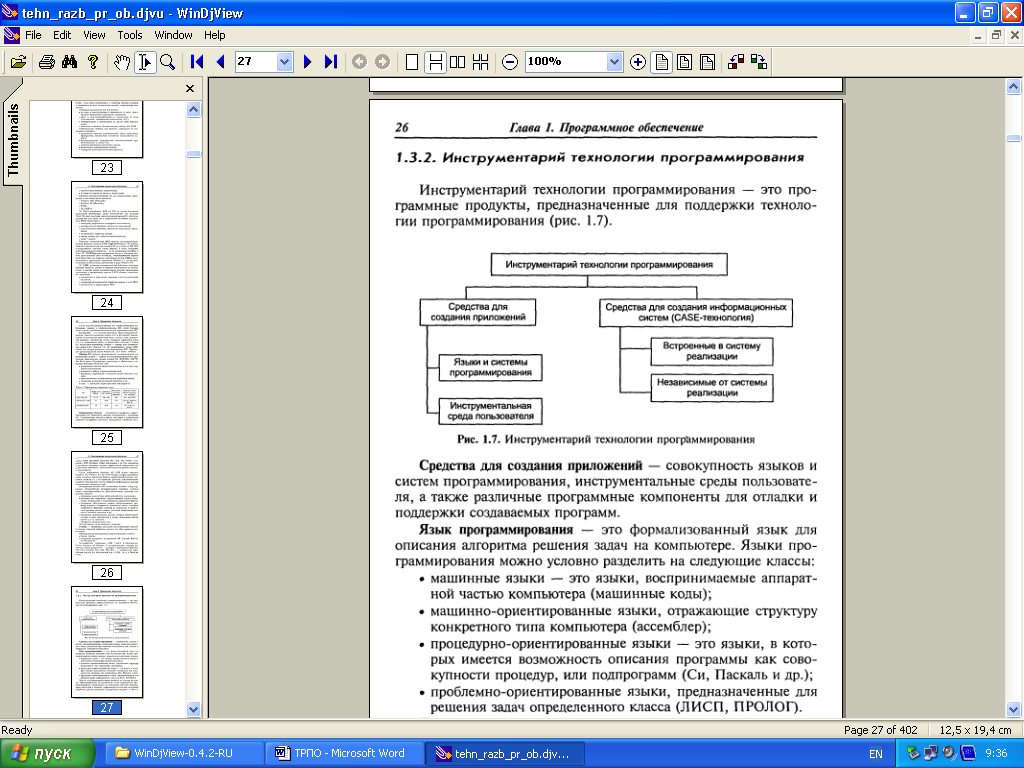


Рисунок – 1. Инструментарий технологии программирования

*Средства для создания приложений* — совокупность языков и систем программирования, инструментальные среды пользователя, а также различные программные компоненты для отладки и поддержки создаваемых программ.

*Язык программирования* — это формализованный язык для описания алгоритма решения задач на компьютере. Языки программирования можно условно разделить на следующие классы:

* машинные языки — это языки, воспринимаемые аппаратной частью компьютера (машинные коды);
* машинно-ориентированные языки, отражающие структуру конкретного типа компьютера (ассемблер);
* процедурно-ориентированные языки — это языки, в которых имеется возможность описания программы как совокупности процедур, или подпрограмм (Си, Паскаль и др.);
* проблемно-ориентированные языки, предназначенные для решения задач определенного класса (ЛИСП, ПРОЛОГ).

Другой классификацией языков является их деление на языки, ориентированные на реализацию основ структурного программирования, основанного на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных различных программных модулей, и объектно-ориентированные языки, поддерживающие понятие объектов, их свойств и методов обработки.

Системы программирования включают:

* компилятор (транслятор);
* интегрированную среду разработки программ (не всегда);
* отладчик;
* средства оптимизации кода программ;
* набор библиотек;
* редактор связей;
* сервисные средства (утилиты) (для работы с библиотеками, текстовыми и двоичными файлами);
* справочные системы;
* систему поддержки и управления продуктами
* программного комплекса.

*Компилятор* транслирует всю программу без ее выполнения. *Трансляторы* (интерпретаторы) выполняют пооперационную обработку и выполнение программы.

*Отладчики (debugger)* — специальные программы, предназначенные для трассировки и анализа выполнения других программ.

*Трассировка* — это обеспечение выполнения в пооператорном варианте.

*Инструментальная среда пользователя* — это специальные средства, встроенные в пакеты прикладных программ, такие как:

* библиотека функций, процедур, объектов и методов обработки;
* макрокоманды;
* клавишные макросы;
* языковые макросы;
* конструкторы экранных форм и объектов;
* генераторы приложений;
* языки запросов высокого уровня;
* конструкторы меню и др.

Интегрированные среды разработки программ объединяют набор средств для их комплексного применения на технологических этапах создания программы. Средства для создания информационных систем (ИС) и технологий поддерживают полный цикл проектирования сложной информационной системы или технологии от исследования объекта автоматизации до оформления проектной и прочей документации на информационную систему или технологию. Они позволяют вести коллективную работу над проектом за счет возможности работы в локальной сети, экспорта-импорта любых фрагментов проекта, организации управления проектом.

Одним из современных средств разработки ИС является CASE-технология (CASE — Computer-Aided System Engineering) — программный комплекс, автоматизирующий весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем.

Средства CASE-технологий делятся:

* на встроенные в систему реализации — все решения по проектированию и реализации привязки к выбранной СУБД;
* независимые от системы реализации — все решения по проектированию ориентированы на унификацию (определение) начальных этапов жизненного цикла программы и средств их документирования, обеспечивают большую гибкость в выборе средств реализации.

Основное достоинство CASE-технологии — это поддержка коллективной работы над проектом за счет возможности работы в локальной сети разработчиков, экспорта (импорта) любых фрагментов проекта, организованного управления проектами.

В некоторых CASE-системах поддерживается кодогенерация программ — создание каркаса программ и создание полного продукта.

Примеры программных продуктов для создания приложений: Visual C++, Delphi, Visual Basic и т. д.

**Пакеты прикладных программ**

Классификация пакетов прикладных программ (ППП) приведена на рисунке 2.

Проблемно-ориентированные ППП. Для некоторых предметных областей возможна типизация функций управления, структуры данных и алгоритмов обработки. Это вызвало разработку значительного количества ППП одинакового функционального назначения:

* ППП автоматизированного бухгалтерского учета;
* ППП финансовой деятельности;
* ППП управления персоналом;
* ППП управления производством;
* банковские информационные системы и т. п.

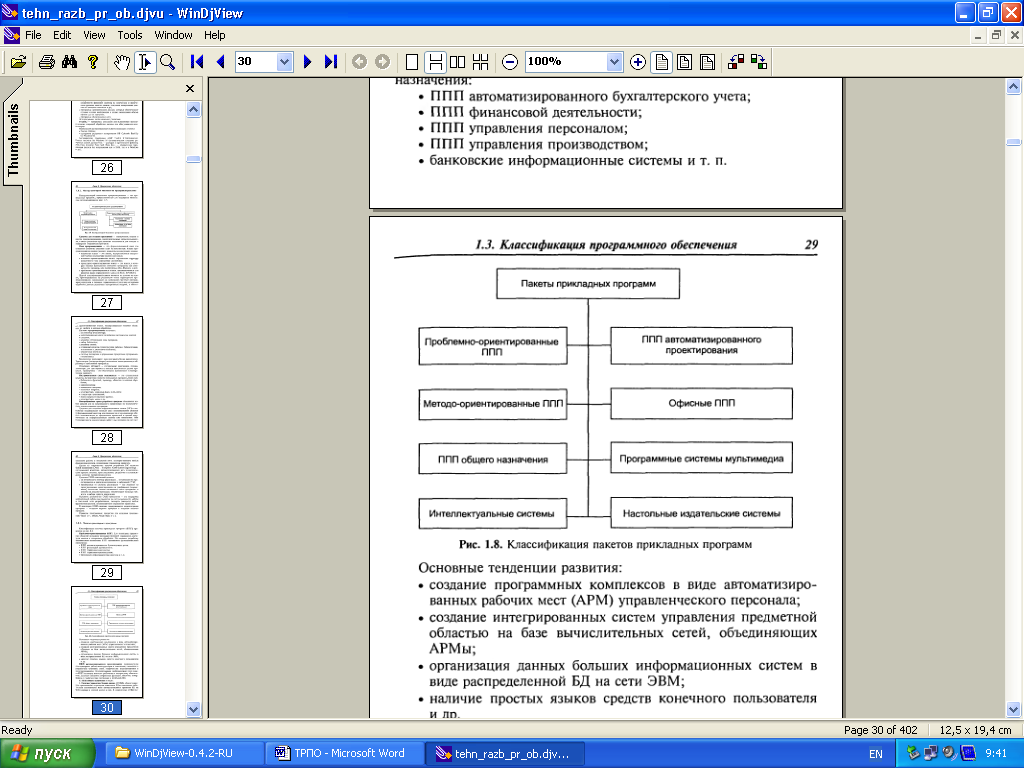


Рисунок - 2. Классификация пакетов прикладных программ

Основные тенденции развития:

* создание программных комплексов в виде автоматизированных рабочих мест (АРМ) управленческого персонала;
* создание интегрированных систем управления предметной областью на базе вычислительных сетей, объединяющих АРМы;
* организация данных больших информационных систем в виде распределенной БД на сети ЭВМ;
* наличие простых языков средств конечного пользователя и др.

ППП автоматизированного проектирования предназначены для поддержки работы конструкторов и технологов, связанных с разработкой чертежей, схем, графическим моделированием и конструированием. Отличительными особенностями этого класса ППП являются высокие требования к аппаратному обеспечению, наличие библиотек встроенных функций, объектов, интерфейсов с графическими системами и БД (AutoCAD).

К ППП общего назначения относят:

1. Системы управления базами данных (СУБД), обеспечивающие организацию и хранение локальных БД на автономно работающих компьютерах либо централизованное хранение БД на файл-сервере и сетевой доступ к ним. В современных СУБД (например, MS Access 2.0) содержатся элементы CASE-технологии процесса проектирования, в частности:
   * визуализирована схема БД;
   * осуществлена автоматическая поддержка целостности БД при различных видах обработки (включение, удаление, модификация);
   * предоставляются так называемые мастера, обеспечивающие поддержки процесса проектирования;
   * созданы шаблоны (прототипы) структур БД, отчетов, форм и т. д.
2. Серверы БД — это ПО, предназначенное для создания и использования при работе в сети интегрированных БД в архитектуре «клиент — сервер». Многопользовательские СУБД в сетевом варианте обработки информации хранят данные на файл-сервере, специально выделенном компьютере, но сама обработка ведется на рабочих станциях. Серверы БД в отличие от этого большую часть обработки (хранение, поиск, извлечение и передачу данных клиенту) данных выполняют самостоятельно, одновременно обеспечивая данными большое число пользователей сети.
3. Общим для различных видов серверов БД является использование реляционного языка SQL (Structured Query Language) для реализации запросов к данным. Большинство серверов БД поддерживает несколько платформ, широкий спектр протоколов передачи данных. Проблемы: обеспечение целостности данных, тиражирование данных по узлам сети и синхронное обновление.
4. Генераторы отчетов (серверы отчетов), обеспечивающие реализацию запросов и формирование отчетов в печатном или экранном виде в условиях сети с архитектурой «клиент — сервер». Сервер отчетов подключается к серверу БД, использующему драйверы сервиса БД (Crystal Reports, Profit for Windows).
5. Текстовые процессоры, предназначенные для работы с текстовыми документами. Развитием данного направления являются издательские системы (Microsoft Word).
6. Табличные процессоры, являющиеся удобной средой для вычислений конечным пользователем, содержат средства деловой графики, средства специализированной обработки (Microsoft Excel).
7. Средства презентационной графики — специализированные программы, предназначенные для создания изображений и их показа на экране, подготовки слайд-фильмов, мультфильмов и их проектирования (Microsoft PowerPoint).
8. Интегрированные пакеты набор нескольких программных продуктов, функционально дополняющих друг друга, поддерживающие единые информационные технологии, реализованные на единой операционной и вычислительной платформе (Microsoft Office). Компоненты интегрированных пакетов могут работать изолированно друг от друга, имеют общий интерфейс, благодаря этому их лучше осваивать.

Методо-ориентированные ППП

Данный класс охватывает программные продукты, обеспечивающие независимо от предметной области и функции информационных систем математические, статистические и другие методы решения задач. Наиболее распространены методы математического программирования, решения дифференциальных уравнений, имитационного моделирования, исследования операций (Storm, SYSTAT, SAS и другие).

Офисные ППП

Данный класс охватывает программы, обеспечивающие ориентационное управление деятельностью офиса:

* органайзеры (планировщики) — ПО для планирования рабочего времени, составления протоколов встреч, расписаний, ведение записей и телефонной книжки. В состав входят: калькулятор, записная книжка, часы, календарь и т. п.
* программы-переводчики, средства проверки орфографии, распознавание текста (Tiger — система распознавания русского языка, Stylus Lingvo Office, содержащий Fine Reader, Stylus for Windows — переводчик на указанный язык, корректор орфографии Lingvo Corrector и резидентный словарь Lingvo);
* коммуникационные пакеты, предназначенные для организации взаимодействия пользователей с удаленными абонентами или информационными ресурсами сети;
* браузеры, средства создания WWW-страниц;
* средства электронной почты (Pegasys Mail).

Настольные издательские системы.

Данный класс ПО включает программы (PageMaker, CorelDraw, PhotoShop for Windows и т. д.), обеспечивающие информационную технологию компьютерной издательской деятельности:

* форматирование и редактирование текстов;
* автоматическую разбивку текста на страницы;
* компьютерную верстку печатной страницы;
* монтирование графики;
* подготовку иллюстраций и т. п.

Программные средства мультимедиа

Основное значение данных программных средств — создание и использование аудио- и видеоинформации для расширения информационного пространства пользователя (различные БД компьютерных произведений искусства, библиотеки звуковых записей и т. д.).

Системы искусственного интеллекта:

* программы-оболочки для создания экспертных систем путем наполнения баз знаний и правил логического вывода;
* готовые экспертные системы для принятия решений в рамках определенных предметных областей;
* системы анализа и распознавания речи, текста и т. п.

Примеры систем искусственного интеллекта: FIDE, MYSIN, Guru и др.

**Контрольные вопросы**

1. Перечислите основные характеристики программ.
2. Приведите существующую классификацию программного обеспечения.
3. Дайте определение и перечислите основные характеристики системного программного обеспечения.
4. Дайте определение и перечислите основные характеристики прикладного программного обеспечения.
5. Дайте определение и охарактеризуйте инструментарий технологии программирования.
6. Назовите основные элементы интегрированной среды разработки Delphi (Delphi IDE)
7. Перечислите поэтапное создание программного кода для каждого практического задания.