История развития языков программирования

Введение

1. Несмотря на огромное количество программных продуктов которыми мы пользуемся может возникнуть необходимость составления (написания) собственной программы.
2. Освоение правил написания программы позволяет лучше ориентироваться в каждой конкретной программе. Понимать что может сделать программа, а чего не может.

Основные понятия языков программирования

- **Язык программирования**

Алгоритмический язык

Programming language; Algorithmic language

Язык программирования - искусственный (формальный) язык, предназначенный для записи алгоритмов.

Язык программирования задается своим описанием и реализуется в виде специальной программы: компилятора или интерпретатора.

Основные понятия языков программирования

- **Декларативный язык программирования От лат.Declaratio - объявление**

Декларативный язык программирования - язык программирования высокого уровня, построенный:
- на описании данных; и
- на описании искомого результата.

Декларативные языки подразделяются на функциональные и логические языки.

Например: ПРОЛОГ

Основные понятия языков программирования

- **Исходный код**

Исходный код - текст программы на алгоритмическом языке. В компьютере исходный текст либо непосредственно выполняется интерпретатором, либо предварительно переводится компилятором в стандартный загрузочный код, способный многократно исполняться в определенной вычислительной среде.

Основные понятия языков программирования

- **Машинный язык**

Машинный язык - язык программирования, элементами которого являются команды компьютера, характеризующиеся:
- количеством операндов в команде;
- назначением информации, задаваемой в операндах;
- набором операций, которые может выполнить компьютер и др.

Конструкции машинного языка интерпретируются непосредственно аппаратурой.

Основные понятия языков программирования

- **Переменная Variable**

Переменная - в языках программирования - именованная часть памяти, в которую могут помещаться разные значения переменной. Причем в каждый момент времени переменная имеет единственное значение. В процессе выполнения программы значение переменной может изменяться.

Тип переменных определяется типом данных, которые они представляют.

Основные понятия языков программирования

- **Процедурно-ориентированный язык программирования. Операторный язык программирования; Императивный язык программирования**

От лат.Imperativus - повелительный

Процедурно-ориентированный язык программирования - язык программирования высокого уровня, в основу которого положен принцип описания (последовательности) действий, позволяющей решить поставленную задачу.

Обычно процедурно-ориентированные языки задают программы как совокупности процедур или подпрограмм.

Примеры: BASIC, PASCAL, FORTRAN, Си

Основные понятия языков программирования

- **Семантика Semantics**

Семантика - в программировании - система правил истолкования отдельных языковых конструкций. Семантика определяет смысловое значение предложений алгоритмического языка.

- **Синтаксис От греч.Syntaxis - порядок**

Синтаксис - набор правил построения фраз алгоритмического языка, позволяющий определить, осмысленные предложения в этом языке.

Основные понятия языков программирования

- **Система программирования**

Система программирования - программная система, предназначенная для разработки программ на конкретном языке программирования. Система программирования предоставляет пользователю специальные средства разработки программ: транслятор, (специальный) редактор текстов программ, библиотеки стандартных подпрограмм, программную документацию, отладчик и др.

Основные понятия языков программирования

- **Тип данных Data type**

Тип данных - характеристика набора данных, которая определяет:
- диапазон возможных значений данных из набора;
- допустимые операции, которые можно выполнять над этими значениями;
- способ хранения этих значений в памяти.

Различают:
- простые типы данных: целые, действительные числа и др.;
- составные типы данных: массивы, файлы и др.

Основные понятия языков программирования

- **Язык ассемблера Ассемблер**

Assembly language; Assembler

Язык ассемблера - согласно ГОСТ 19781-90 - язык программирования; символьная форма машинного языка с рядом возможностей, характерных для языка высокого уровня. Обычно язык ассемблера включает макросредства.

Основные понятия языков программирования

- **Язык высокого уровня Язык высокого уровня - согласно ГОСТ 19781-90 - язык программирования, понятия и структура которого удобны для восприятия человеком.**

Языки высокого уровня отражают потребности программиста, но не возможности системы обработки данных.

Основные понятия языков программирования

- **Жизненный цикл программного обеспечения - период разработки и эксплуатации программного обеспечения, в котором обычно выделяют этапы:**
  -1- возникновение и исследование идеи;
  -2- анализ требований и проектирование;
  -3- программирование;
  -4- тестирование и отладка;
  -5- ввод программы в действие;
  -6- эксплуатация и сопровождение;
  -7- завершение эксплуатации.

Основные понятия языков программирования

- **Дистрибутивная система**

От англ.Distribute - распределять

Дистрибутивная система - набор файлов, позволяющий инсталлировать рабочий вариант программной системы.

- **Инсталляция**

Установка

Installation

Инсталляция - процесс установки программного продукта на конкретную машину, для конкретного пользователя.

Инсталляция проводится с помощью специальной программы поставляемой разработчиком.

Обзор языков программирования

- **Ассемблер**
- К языкам низкого уровня относятся языки Ассемблера. Свое название они получили от имени системной программы Ассемблер, которая преобразует исходные программы, написанные на таких языках, непосредственно в коды машинных команд. Частями здесь служат операторы, а результатом сборки последовательность машинных команд Язык Ассемблера объединяет в себе достоинства языка машинных команд и некоторые черты языков высокого уровня. Ассемблер обеспечивает возможность применения символических имен в исходной программе и избавляет программиста от утомительного труда (неизбежного при программировании на языке машинных команд) по распределению памяти компьютера для команд, переменных и констант.
- Ассемблер позволяет также гибко и полно использовать технические возможности компьютера, как и язык машинных команд. Транслятор исходных программ в Ассемблере проще транслятора, требующегося для языка программирования высокого уровня. На Ассемблере можно написать столь же эффективную по размеру и времени выполнения программу, как и программу на языке машинных команд. Это достоинство отсутствует у языков высокого уровня. Этот язык часто применяют для программирования систем реального времени, технологическими процессами и оборудованием, обеспечение работы информационно-измерительных комплексов. К таким системам обычно предъявляются высокие требования по объему занимаемой машинной памяти. Часто язык Ассемблера дополняется средствами формирования макрокоманд, каждая из которых эквивалентна целой группе машинных команд. Такой язык называют языком макроассемблера. Применение мак "строительных" блоков и приближает язык Ассемблера к языку высокого уровня. Ассемблер машинно-зависимый язык, т. е. он отражает особенности архитектуры конкретного типа компьютера

Обзор языков программирования

- **Паскаль**

Язык программирования Паскаль был разработан профессором кафедры вычислительной техники Швейцарского Федерального института технологии Николасом Виртом в 1968 году как альтернатива существующим и все усложняющимся языкам программирования, таким, как PL/1, Algol, Fortran. Интенсивное развитие Паскаля привело к появлению уже в 1973 году его стандарта в виде пересмотренного сообщения, а число трансляторов с этого языка в 1979 году перевалило за 80. В начале 80-х годов Паскаль еще более упрочил свои позиции с появлением трансляторов MS-Pascal и Turbo-Pascal для ПЭВМ. С этого времени Паскаль становится одним из наиболее важных и широко используемых языков программирования. Существенно то, что язык давно вышел за рамки академического и узко профессионального интереса и используется в большинстве университетов высокоразвитых стран не только как рабочий инструмент пользователя. Важнейшей особенностью Паскаля является воплощенная идея структурного программирования. Другой существенной особенностью является концепция структуры данных как одного из фундаментальных понятий.

Основные причины популярности Паскаля заключаются в следующем:

- простота языка позволяет быстро его освоить и создавать алгоритмически сложные программы

- развитые средства представления структур данных обеспечивают удобство работы как с числовой, так и с символьной и битовой информацией

- наличие специальных методик создания трансляторов с Паскаля упростило их разработку и способствовало широкому распространению языка

- оптимизирующие свойства трансляторов с Паскаля позволяют создавать эффективные программы. Это послужило одной из причин использования Паскаля в качестве языка системного программирования

- в языке Паскаль реализуются идеи структурного программирования, что делает программу наглядной и дает хорошие возможности для разработки и отладки

Обзор языков программирования

- **Си**
- Сотрудник фирмы Bell Labs Денис Ритчи создал язык Си в 1972 году во время совместной работы с Кеном Томпсоном, как инструментальное средство для реализации операционной системы Unix, однако популярность этого языка быстро переросла рамки конкретной операционной системы и конкретных задач системного программирования. В настоящее время любая инструментальная и операционная система не может считаться полной если в ее состав не входит компилятор языка Си. Ритчи не выдумывал Си просто из головы – прообразом служил язык Би разработанный Томпсоном. Язык программирования Си был разработан как инструмент для программистов-практиков. В соответствии с этим главной целью его автора было создание удобного и полезного во всех отношениях языка.
- Си является орудием системного программиста и позволяет глубоко влезать в самые тонкие механизмы обработки информации на ЭВМ. Хотя язык требует от программиста высокой дисциплины, он не строг в формальных претензиях и допускает краткие формулировки.
- Си – современный язык. Он включает в себя те управляющие конструкции, которые рекомендованы теорией и практикой программирования. Его структура побуждает программиста использовать в своей работе нисходящее проектирование, структурное программирование и пошаговую разработку модулей.
- Си – мощный и гибкий язык. Большая часть операционной системы Unix, компиляторы и интерпретаторы языков Фортран, Паскаль, Лисп, и Бейсик написаны именно с его помощью.
- Си – удобный язык. Он достаточно структурирован, чтобы поддерживать хороший стиль программирования и вместе с тем не связан жесткими ограничениями. В некотором смысле язык Си – самый универсальный, т.к. кроме набора средств, присущих современным языкам программирования высокого уровня (структурность, модульность, определенные типы данных), в него включены средства для программирования практически на уровне ассемблера. Большой набор операторов и средств требуют от программиста осторожности, аккуратности и хорошего знания языка со всеми его преимуществами и недостатками.

Обзор языков программирования

- **Си++**
- Язык C++ появился в начале 80-х годов. Созданный Бьерном Страуструпом с первоначальной целью избавить себя и своих друзей от программирования на ассемблере, Си или различных других языках высокого уровня.
- По мнению автора языка, различие между идеологией Си и C++ заключается примерно в следующем: программа на Си отражает “способ мышления” процессора, а C++ - способ мышления программиста. Отвечая требованиям современного программирования, C++ делает акцент на разработке новых типов данных наиболее полно соответствующих концепциям выбранной области знаний и задачам приложения. Класс является ключевым понятием C++. Описание класса содержит описание данных, требующихся для представления объектов этого типа и набор операций для работы с подобными объектами.
- В отличие от традиционных структур Си и Паскаля, членами класса являются не только данные, но и функции. Функции – члены класса имеют привилегированный доступ к данным внутри объектов этого класса и обеспечивают интерфейс между этими объектами и остальной программой. При дальнейшей работе совершенно не обязательно помнить о внутренней структуре класса и механизме работы встроенных функций. В этом смысле класс подобен электрическому прибору – мало кто знает о его устройстве, но все знают, как им пользоваться.
- Язык C++ является средством объектного программирования, новейшей методики проектирования и реализации программ, которая в текущем десятилетии, скорее всего, заменит традиционное процедурное программирование. Главной целью создателя языка доктора Бьерна Страустрапа было оснащение языка C++ конструкциями, позволяющими увеличить производительность труда программистов и облегчить процесс овладения большими программными продуктами.

Обзор языков программирования

- **Кобол**
- Кобол - это сравнительно старый язык, разработанный прежде всего для исследований в экономической сфере. Язык позволяет эффективно работать с большим количеством данных, он насыщен разнообразными возможностями поиска, сортировки и распределения. О программах на Коболе, основанных на широком использовании английского языка, говорят, что они понятны даже тем, кто не владеет Коболом, поскольку тексты на этом языке программирования не нуждаются в каких-либо специальных комментариях. Подобные программы принято называть самодокументирующимися. К числу других плюсов Кобола обычно относят его структурированность. Довольно мощные компиляторы с этого языка разработаны для персональных компьютеров. Некоторые из них столь эффективны, что программу, отлаженную на персональном компьютере, нетрудно перенести на большие ЭВМ.
- Перечисляя минусы нельзя не вспомнить о том, что на Коболе можно запрограммировать лишь простейшие алгебраические вычисления. Для инженерных расчетов этот язык не годится. Еще одна причина, которая в какой-то мере сдерживает развитие языка, - это наличие в США специально созданного отраслевого комитета, вырабатывающего стандарты, за соблюдением которых следит правительственная комиссия. Как это всегда бывает в подобных случаях, фирмы, занимающиеся разработкой программного обеспечения, не торопятся подгонять свои заготовки к жестким требованиям комиссии, отсутствует конкуренция версий, а в итоге проигрывает распространение языка

Обзор языков программирования

- **Бейсик**
- Бейсик (Basic - Beginner's All-Purpose Symbolic Instruction Code – "универсальный символический код инструкций для начинающих"). Прямой потомок Фортрана и до сих пор самый популярный язык программирования для персональных компьютеров. Появился Бейсик в 1963 году (назвать автора было бы трудно, но основная заслуга в его появлении несомненно принадлежит американцам Джону Кемени и Томасу Курцу). Как и любые преимущества, простота Бейсика оборачивалась, особенно в ранних версиях трудностями структурирования; кроме того, Бейсик не допускал рекурсию – интересный прием, позволяющий составлять эффективные и в то же время короткие программы.
- Разработаны мощные компиляторы Бейсика, которые обеспечивают не только богатую лексику и высокое быстродействие, но и возможность структурного программирования. По мнению некоторых программистов, наиболее интересными версиями являются GWBASIC, Turbo-Basic и Quick Basic.
- В свое время появление Quick Basic ознаменовало рождение второго поколения систем программирования на языке Бейсик. Он предоставлял возможность модульного и процедурного программирования, создания библиотек, компиляции готовых программ и прочее, что вывело его на уровень таких классических языков программирования, как Си, Паскаль, Фортран и др. Более того, в связи с отсутствием официального стандарта языка Бейсик, его реализация в виде Quick Basic стала фактическим стандартом. Безусловными лидерами среди различных версий Бейсика были Quick Basic 4.5 и PDS 7.1 фирмы Microsoft, появившиеся в конце 80-х годов.

Обзор языков программирования

- **Лисп**
- Язык Лисп был предложен Дж. Маккарти в работе в 1960 году и ориентирован на разработку программ для решения задач не численного характера. Английское название этого языка – LISP является аббревиатурой выражения LISt Processing (обработка списков) и хорошо подчеркивает основную область его применения. Понятие “список” оказалось очень емким. В виде списков удобно представлять алгебраические выражения, графы, элементы конечных групп, множества, правила вывода и многие другие сложные объекты. Списки являются наиболее гибкой формой представления информации в памяти компьютеров. Неудивительно поэтому, что удобный язык, специально предназначенный для обработки списков, быстро завоевал популярность.
- После появления Лиспа различными авторами был предложен целый ряд других алгоритмических языков ориентированных на решение задач в области искусственного интеллекта, среди которых можно отметить Плэнер, Снобол, Рефал, Пролог. Однако это не помешало Лиспу остаться наиболее популярным языком для решения таких задач. На протяжении почти сорокалетней истории его существования появился ряд диалектов этого языка: Common LISP, Mac LISP, Inter LISP, Standard LISP и др. Различия между ними не носят принципиального характера и в основном сводятся к несколько отличающемуся набору встроенных функций и некоторой разнице в форме записи программ. Поэтому программист, научившийся работать на одном из них без труда сможет освоить и любой другой. Большим достоинством Лиспа является его функциональная направленность, т. е. программирование ведется с помощью функций. Причем функция понимается как правило, сопоставляющее элементам некоторого класса соответствующие элементы другого класса. Сам процесс сопоставления не оказывает никакого влияния на работу программы, важен только его результат – значение функции. Это позволяет относительно легко писать и отлаживать большие программные комплексы. Ясность программ, четкое разграничение их функций, отсутствие каверзных побочных эффектов при их выполнении является обязательными требованиями к программированию таких логически сложных задач, каковыми являются задачи искусственного интеллекта. Дисциплина в программировании становится особенно важной, когда над программой работает не один человек, а целая группа программистов.
- Язык программирования Лисп предназначен в первую очередь для обработки символьной информации. Поэтому естественно, что в мире Лиспа числа играют далеко не главную роль. Основные типы данных в Лиспе называются “атом” и “точечная пара”.

Обзор языков программирования

- **Пролог**
- Язык логического программирования предназначен для представления и использования знаний о некоторой предметной области. Программы на этом языке состоят из некоторого множества отношений, а ее выполнение сводится к выводу нового отношения на основе заданных. В Прологе реализован декларативный подход, при котором достаточно описать задачу с помощью правил и утверждений относительно заданных объектов. Если это описание является достаточно точным, то ЭВМ может самостоятельно найти требуемое решение.

Обзор языков программирования

- **Object PAL**
- Object PAL является мощным языком программирования. Object PAL представляет собой объектно-ориентированный, управляемый по событиям, визуальный язык программирования. На начальном уровне функциональности Object PAL можно осуществлять операции с данными, создавать специальные меню, а также управлять сеансом ввода данных. События в Object PAL порождают команды, которые имитируют эффект использования Paradox в интерактивном режиме. Существует возможность автоматизировать часто выполняемые задания, а также осуществлять над таблицами, формами и отчетами действия, которые были не доступны при интерактивной работе. Также Object PAL предоставляет все средства полнофункционального языка программирования в среде Windows. Можно использовать Object PAL для создания законченных систем, в которых реализованы специальная система меню, справочная система, а также всевозможные проверки данных. В Object PAL можно сохранить свои наработки в динамически компонуемой библиотеке, доступ к которой будут иметь несколько форм. Кроме того, можно установить связь с другими динамическими библиотеками, содержащие прграммы написанные на таких языках как Си, C++ или Паскаль.
- Object PAL может быть использован в качестве инструмента для создания автономных программ. Можно написать законченное Windows–приложение и запустить его под Paradox.
- Object PAL поддерживает механизм динамического обмена данными в качестве как клиента, так и сервера. Кроме того, Object PAL поддерживает в качестве клиента механизм работы с составными документами. В дополнение к сказанному существует возможность включать в свое приложение мультимедийные средства, снабдив выполняемое приложение звуковыми и анимационными эффектами.

Обзор языков программирования

- **Java**
- Язык Java зародился как часть проекта создания передового программного обеспечения (ПО) для различных бытовых приборов. Реализация проекта была начата на языке C++, но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента - языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами. Язык Java потребовался для создания интерактивных продуктов для сети Internet. Фактически, большинство архитектурных решений, принятых при создании Java, было продиктовано желанием предоставить синтаксис, сходный с Си и Си++. В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода. В Java добавлены все хорошие черты C++.
- Три ключевых элемента объединились в технологии языка Java
- - Java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML.
- - Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты.
- - Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов.

Обзор языков программирования

- **Языки программирования для компьютерных сетей**
- Языки программирования для компьютерных сетей являются интерпретируемыми. Интерпретаторы для них распространяются бесплатно, а сами программы – в исходных текстах. Такие языки называются скрипт – языками.
- **Perl**

Perl является интерпретируемым языком, созданным программистом Лари Уоллом для обработки больших текстов и файлов и расшифровывается, как Practical Extraction and Report Language (язык для практического извлечения данных и составления отчетов). С помощью Perl вы, например, можете создать скрипт, который открывает один или несколько файлов, обрабатывает информацию и записывает результаты.

Perl - язык, приспособленный для обработки произвольных текстовых файлов, извлечения из них необходимой информации и выдачи сообщений. Perl также удобен для написания различных системных программ. Этот язык прост в использовании, эффективен, но про него трудно сказать, что он элегантен и компактен. Синтаксис выражений Perl близок к синтаксису Си. Рекурсия может быть произвольной глубины. Хотя Perl приспособлен для сканирования текстовых файлов, он может обрабатывать так же двоичные данные. Perl позволяет использовать регулярные выражения, создавать объекты, вставлять в программу на Си или Си++ куски кода на Perl, а также позволяет осуществлять доступ к базам данных, в том числе Oracle.

С изобретением World Wide Web, Perl оказался прекрасным средством для взаимодействия с web-серверами через Common Gateway Interface (CGI) - общий интерфейс взаимодействия. Команды Perl могут легко получить данные из формы HTML или другого источника и выполнить с ними какое-нибудь действие.

Язык **PHP** (1995-1997гг) обладает средствами доступа к БД и используется создателями

Обзор языков программирования

- **Учебный язык программирования — язык программирования**, предназначенный для обучения специалистов программированию. Такой язык должен отвечать главному требованию: простота. Чем проще он будет, тем быстрее его освоит новичок. Возможности таких языков могут быть ниже чем возможности полноценных, но они не предназначены для серьёзной работы. Однако, такие языки тоже способны к развитию: многие учебные языки программирования впоследствии превратились в полноценные языки высокого уровня, например Паскаль, BASIC, Пролог.

Структурное программирование

- **Структурное программирование** — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 70-х годах XX века Э. Дейкстрой, разработана и дополнена Н. Виртом.
- В соответствии с данной методологией

1. Любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:
    1. **последовательное исполнение** — однократное выполнение операций в том порядке, в котором они записаны в тексте программы;
    2. **ветвление** — однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;
    3. **цикл** — многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла).

Структурное программирование

- В программе базовые конструкции могут быть вложены друг в друга произвольным образом, но никаких других средств управления последовательностью выполнения операций не предусматривается.
- Повторяющиеся фрагменты программы (либо не повторяющиеся, но представляющие собой логически целостные вычислительные блоки) могут оформляться в виде т. н. подпрограмм (процедур или функций). В этом случае в тексте основной программы, вместо помещённого 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.
- Разработка программы ведётся пошагово, методом «сверху вниз».
- Сначала пишется текст основной программы, в котором, вместо каждого связного логического фрагмента текста, вставляется вызов подпрограммы, которая будет выполнять этот фрагмент. Вместо настоящих, работающих подпрограмм, в программу вставляются «заглушки», которые ничего не делают. Полученная программа проверяется и отлаживается.

Структурное программирование

- После того, как программист убедится, что подпрограммы вызываются в правильной последовательности (то есть общая структура программы верна), подпрограммы-заглушки последовательно заменяются на реально работающие, причём разработка каждой подпрограммы ведётся тем же методом, что и основной программы. Разработка заканчивается тогда, когда не останется ни одной «затычки», которая не была бы удалена. Такая последовательность гарантирует, что на каждом этапе разработки программист одновременно имеет дело с обозримым и понятным ему множеством фрагментов, и может быть уверен, что общая структура всех более высоких уровней программы верна. При сопровождении и внесении изменений в программу выясняется, в какие именно процедуры нужно внести изменения, и они вносятся, не затрагивая части программы, непосредственно не связанные с ними. Это позволяет гарантировать, что при внесении изменений и исправлении ошибок не выйдет из строя какая-то часть программы, находящаяся в данный момент вне зоны внимания программиста.

Модульное программирование

- **Модуль** в программировании представляет собой функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом или поименованной непрерывной его части (например, Active Oberon), предназначенный для использования в других программах. Модули позволяют разбивать сложные задачи на более мелкие, в соответствии с принципом модульности. Обычно проектируются таким образом, чтобы предоставлять программистам удобный для многократного использования функционал (интерфейс) в виде набора функций, классов, констант. Модули могут объединяться в пакеты и, далее, в библиотеки.
- Модули могут быть *обычными*, т.е. написанными на том же языке, что и программа, в которой они используются, либо *модулями расширения*, которые пишутся на отличном от языка основной программы языке. Модули расширения обычно пишутся на более низкоуровневом языке, что позволяет получить выигрыш в скорости выполнения (производительности) программы.

Отладка программы

- **Отладка** — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, приходится :
- узнавать текущие значения переменных;
- и выяснять, по какому пути выполнялась программа.
- Существуют две взаимодополняющие технологии отладки.
- Использование отладчиков — программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.
- Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода — на экран, принтер, громкоговоритель или в файл. Вывод отладочных сведений в файл называется **журналированием**.

Тестирование программ

- **По объекту тестирования:**
- Функциональное тестирование (functional testing)
- Нагрузочное тестирование
  - Тестирование производительности (perfomance/stress testing)
  - Тестирование стабильности (stability/load testing)
- Тестирование удобства использования (usability testing)
- Тестирование интерфейса пользователя (UI testing)
- Тестирование безопасности (security testing)
- Тестирование локализации (localization testing)
- Тестирование совместимости (compatibility testing)

Тестирование программ

- **По знанию системы:**
- Тестирование чёрного ящика (black box)
- Тестирование белого ящика (white box)
- Тестирование серого ящика (gray box)
- **По степени автоматизированности:**
- Ручное тестирование (manual testing)
- Автоматизированное тестирование (automated testing)
- Полуавтоматизированное тестирование (semiautomated testing)

Тестирование программ

- **По степени изолированности компонентов:**
- Компонентное (модульное) тестирование (component/unit testing)
- Интеграционное тестирование (integration testing)
- Системное тестирование (system/end-to-end testing)
- **По времени проведения тестирования:**
- Альфа тестирование (alpha testing)
  - Тестирование при приёмке (smoke testing)
  - Тестирование новых функциональностей (new feature testing)
  - Регрессионное тестирование (regression testing)
  - Тестирование при сдаче (acceptance testing)
- Бета тестирование (beta testing)