1. ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Оглавление

§1.1 Язык программирования. Классификация языков программирования

Чтобы управлять компьютером, необходимо организовать способ общения с ним. В отличие от животных, которые разговаривают между собой на собственных, непонятных для нас языках, компьютеры понимают языки программирования, созданные людьми.

Компьютерная программа представляет собой набор инструкций, реализующий некоторый алгоритм, записанный в форме, понятной для компьютера. **Алгоритм** представляет собой конечную совокупность точно заданного набора инструкций, определяющего порядок действий исполнителя для решения некоторой задачи. Языки, на котором описываются данные алгоритмы, понятны людям, но отличаются от языков человеческого общения обладают более жесткой организацией и очень ограниченным словарным запасом.

Язык программирования определяет набор лексических, синтаксических и семантических правил, используемых при составлении компьютерной программы.

Лексика — совокупность слов того или иного языка, части языка или слов, которые знает тот или иной человек или группа людей.

Синтаксис — сторона языка программирования, которая описывает структуру программ как наборов символов (обычно говорят — безотносительно к содержанию).

Синтаксису языка противопоставляется его семантика. Синтаксис языка описывает «чистый» язык, в то же время семантика приписывает определенные действия различным синтаксическим конструкциям.

Семантика в программировании – это система правил, определяющих поведение отдельных языковых конструкций (определяет смысловое значение предложений алгоритмического языка).

Существует множество критериев, по которым можно классифицировать языки программирования. Частые варианты классификации представлены ниже:

1. **По парадигме** (декларативные, императивные, структурированные и т.п.);
2. **По системе типов** (динамические, статические, сильно- и слаботипизированные, нетипизированные и т.п.);
3. **По уровню абстракции** (высокого, низкого уровня);
4. **По модели исполнения** (компилируемые, интерпретируемые).

Четкой классификации не существует, по той простой причине, что существуют буквально тысячи ЯП, и в любой категории классификации обнаруживается практически непрерывный спектр.

На рисунке 1.1 представлена схема классификации ЯП на процедурные и непроцедурные.

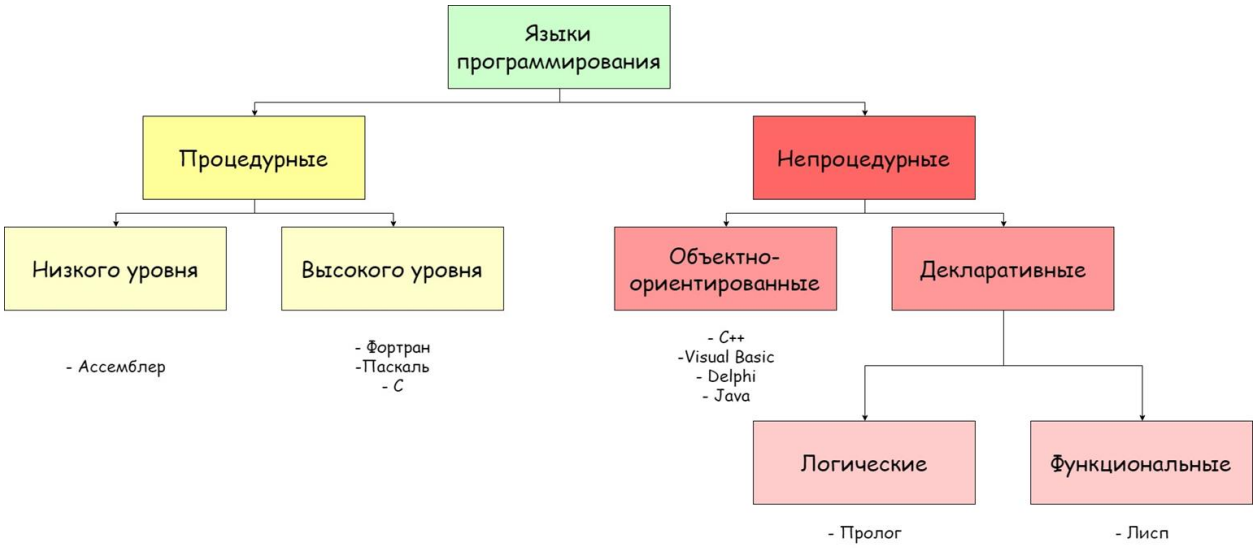

Рисунок 1.1 Классификая языков программирования по уровню абстракции

Императивное программирование — это парадигма программирования (стиль написания исходного кода компьютерной программы), для которой характерно следующее:

- в исходном коде программы записываются инструкции (команды);
- инструкции должны выполняться последовательно;
- данные, получаемые при выполнении предыдущих инструкций, могут читаться из памяти последующими инструкциями;
- данные, полученные при выполнении инструкции, могут записываться в память.

Императивная программа похожа на приказы, выражаемые повелительным наклонением в естественных языках, то есть представляют собой последовательность команд, которые должен выполнить компьютер.

Декларативное программирование — это парадигма программирования, в которой задаётся спецификация решения задачи, то есть описывается, что представляет собой проблема и ожидаемый результат.

Процедурное программирование — программирование на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка.

Процедурные языки разделяют на **языки низкого** и **высокого уровня**.

Язык низкого уровня – это язык программирования, предназначенный для определенного типа компьютера и отражающий его внутренний машинный код.

Языки низкого уровня часто называют машинно-ориентированными языками. Их сложно конвертировать для использования на компьютерах с разными центральными процессорами, а также довольно сложно изучать, поскольку для этого требуется хорошо знать внутренние принципы работы компьютера.

Язык высокого уровня – это язык программирования, предназначенный для удовлетворения требований программиста, он не зависит от внутренних машинных кодов компьютера любого типа.

Языки высокого уровня используют для решения проблем, и поэтому их часто называют проблемно-ориентированными языками. Каждая команда языка высокого уровня эквивалентна нескольким командам в машинных кодах, поэтому программы, написанные на языках высокого уровня, более компактны, чем аналогичные программы в машинных кодах.

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

В объектно-ориентированных языках не описывают подробной последовательности действий для решения задачи, хотя они содержат элементы

процедурного программирования. Программа пишется в терминах объектов, которые обладают свойствами и поведением. Объекты обмениваются сообщениями.

Примеры объектно-ориентированных языков: Object Pascal, C++, C#, Java, JavaScript, Objective-C, PHP, Python.

Функциональное программирование – методология программирования, где программа описывает вычисление некоторой функции. Обычно эта функция задается как композиция других, более простых, те в свою очередь разлагаются на еще более простые и т.д.

Один из основных элементов в функциональных языках – рекурсия, т.е. вычисление значения функции через значение этой же функции от других элементов. Присваивания и циклов в классических функциональных языках нет. Представителями этой группы являются ЛИСП, ML и Haskell.

Логическое программирование — парадигма программирования, основанная на автоматическом доказательстве теорем.

В логических языках программа вообще не описывает действий. Она задает данные и соотношения между ними. После этого системе можно задавать вопросы. Машина перебирает известные и заданные в программе данные и находит ответ на вопрос. Порядок перебора не описывается в программе, а неявно задается самим языком. Классическим языком логического программирования считается ПРОЛОГ. Построение логической программы вообще не требует алгоритмического мышления, программа описывает статические отношения объектов, а динамика находится в механизме перебора и скрыта от программиста.

§1.2 История развития языков программирования

Первым языком программирования можно считать так называемый **машинный код**, который представлял собой набор нулей и единиц, совмещенных в определенной последовательности. Нулю соответствовало отсутствие напряжения в определенном узле ЭВМ, единице – подача напряжения на некоторый момент времени. Запись с помощью этого языка была очень сложной и утомительной.

Для облегчения работы программистов в конце 1940-х гг. был разработан язык **ассемблер**. Вместо двоичных цифр, которые обозначали какую-либо команду, записывались короткие слова или аббревиатуры. Программисты считают ассемблер

языком программирования низкого уровня, поскольку он близок к языку самого низкого уровня – машинному. Программы, написанные на ассемблере, напрямую зависят от характеристик конкретного процессора, поэтому его называют **машинно-ориентированным языком**. Написание программ на ассемблере является довольно сложной задачей, к тому же необходимы знания устройств компьютера. И тем не менее программы на ассемблере – самые эффективные и работоспособные.

Приблизительно в 50-е годы американские ученые разработали довольно успешный язык **Ada,** который использовался для управления военной техникой. Также очень большую роль в ИТ-индустрию внес язык под названием **Fortran**. Он возник в глубинах компании IBM – главного компьютерного гиганта того времени, и активно использовался для решения научных и технических задач.

Европейские ученые не на шутку взволновались успехами американцев и решили создавать свой язык программирования, дабы не дать США доминировать в области программирования. Благодаря их решению история развития языков программирования дополнилась еще одним представителем – языком **Algol**, который решал примерно те же задачи, что и его американский конкурент.

Один из участников работы над упомянутым выше Алголом по имени Никлаус Вирт решил создать более универсальный и продвинутый язык. В итоге он представил миру такой легендарный язык программирования, как **Паскаль**. Именно он внес существенный вклад в развитие данной области знаний и послужит прочной основой для появления других, более совершенных языков. Паскаль стал одним из первых языков, использующих структурное программирование, довольно простой и легко запоминающийся синтаксис. В будущем многие компании и индивидуальные программисты создавали на базе Паскаля свои варианты языков. К примеру, известная Apple создала расширение Паскаля под названием Object Pascal, а компания Borland – очень популярную и удачную интегрированную среду разработки Turbo Pascal.

В 70-х годах велись активные разработки языка C, который в будущем послужил надежной платформой для создания целого ряда своих более совершенных вариантов: Си Шарп, C++ и других. C был уже полноценным высокоуровневым языком программирования, на котором возможно реализовывать практически любые задачи по созданию самого разнообразного ПО. Известный и популярный в наше время язык

Objective-C, который разработан компанией Apple и активно используется для написания программного обеспечения на их гаджеты и другие продукты, создан именно на основе того самого языка С из далеких 70-х.

История создания языков программирования была бы не полной, если не упоминать еще и о таких важнейших языках, как Java, PHP, HTML. Java возник в середине 90-х годов и сразу получил широкое применение и популярность. с его помощью одинаково легко пишутся как программы на ПК, так и различные скрипты, веб-приложения и многое другое. HTML язык был разработан британским программистом Т. Бернерсом-Ли в начале 90-х. именно он стал основой всей сети интернет и имеющихся сейчас в не миллионов сайтов. Что касается PHP, то этот популярный нынче язык также возник в 1995 году, имел открытый исходный код и способен реализовать практически любую задумку в сфере создания динамических вебсайтов.

C++ был изобретен Бьярном Страуструпом в 1979 году в Bell Laboratories. Сначала Страуструп назвал новый язык «С с классами». однако в 1983 году язык получил название C++.

Страуструп построил C++ на базе С, сохранив все возможности С, его характерные особенности и достоинства. Он перенял также и базовую стратегию С, согласно которой программист, а не язык определяет структуру программу. Важно понимать, что Страуструп не создал совершенно новый язык. Он расширил уже имеющийся весьма успешный язык.

Большая часть средств, добавленных Страуструпом к С, предназначалась для поддержки объектно-ориентированного программирования. В силу того, что C++ перенял эффективность С, его часто используют для разработки высокопроизводительного системного обеспечения.

Язык Java был разработан фирмой Sun Microsystem, C# - фирмой Microsoft. C++ является родительским языком по отношению к этим двум более новым языкам программирования. Синтаксис всех этих трех языков практически идентичен, несмотря на то, что были добавлены, удалены или модифицированы некоторые свойства. Общая осталась для этих языков и объектная модель.

Java и C# создавались для обеспечения более специфических требований, возникающих при написании программ, работающих в сети Интернет. Интернет объединяет много разных типов процессоров и операционных систем. Данные языки позволили создавать межплатформенные, переносимые программы, работающие в различных вычислительных средах.

Переносимость программ, написанных на Java и C#, объясняется тем, что программы компилируются не в машинный, а в промежуточный язык. В случае Java – это **байт-код**. В случае C# он называется **CIL** (Common Intermediate Language). В обоих случая промежуточный код выполняется специальной исполняющей системой. Java-программы используют виртуальную машину Java (Java Virtual Machine, JVM). Для C# программ соответствующая система носит название Common Language Runtime (CLR).

В случае C++ компилятор формирует машинный код, который затем непосредственно выполняется центральным процессором. Если необходимо будет запустить C++-программу в другой системе, необходимо будут перекомпилировать исходный текст программы в машинный код, предназначенный для выбранной вами среды.

§1.3 Системы программирования

Система программирования – это набор специализированных программных продуктов, которые являются инструментальными средствами разработчика. Программные продукты данного класса поддерживают все этапы процесса программирования, отладки и тестирования создаваемых программ.

Система программирования включает следующие программные компоненты:

1. редактор кода;
2. транслятор с соответствующего языка;
3. компоновщик (редактор связей);
4. отладчик;
5. библиотеки подпрограмм.

Редактор текста – это программа для ввода и модификации текста. Данные программы ускоряют процесс разработки и повышают качество программного кода.

Известные редакторы текста – Sublime Text, Atom, Notepad++ и другие.

Транслятор – это программа, предназначенная для преобразования программ, написанных на языках программирования, в программы на машинном языке.

Программа, подготовленная на каком-либо языке программирования, называется **исходным модулем**. В качестве входной информации трансляторы применяют исходные модули и формируют в результате своей работы **объектные модули**, являющиеся входной информацией для редактора связей. Объектный модуль содержит текст программы на машинном языке и дополнительную информацию, обеспечивающую настройку модуля по месту его загрузки и объединение этого модуля с другими независимо оттранслированными модулями в единую программу.

Трансляторы делятся на два класса: **компиляторы** и **интерпретаторы**. **Компиляторы** переводят весь исходный модуль на машинный язык. **Интерпретатор** последовательно переводит на машинный язык и выполнят операторы исходного модуля.

Преимущество интерпретатора перед компилятором состоит в том, что программа пользователя имеет одно представление - в виде текста. При компиляции одна и та же программа имеет несколько представлений - в виде текста и в виде выполняемого файла.

Программа обычно состоит из нескольких отдельных частей, которые часто разрабатываются разными людьми. Например, программа "Hello, World!" состоит из части, которую написали вы, и частей стандартной библиотеки языка C++. Эти отдельные части (иногда называемые единицами трансляции) должны быть скомпилированы, а файлы с результирующим объектным кодом должны быть скомпонованы в единое целое, образуя **выполняемый файл**. Программа, связывающая эти части в одно целое, называется компоновщиком или редактором связей.

Компоновщик, или **редактор связей** - системная обрабатывающая программа, редактирующая и объединяющая объектные (ранее оттранслированные) модули в единые загрузочные, готовые к выполнению программные модули. **Загрузочный модуль** может быть помещен ОС в основную память и выполнен.

Библиотека в программировании (Library) – это сборник подпрограмм или объектов, используемых для разработки программ. Можно сказать, и так: это набор классов, компонентов или модулей для разных задач.

В программах много стандартных элементов, например, кнопки, проигрыватели видео, запросы и т. д. Нет необходимости каждый раз писать их с нуля, потому что все это уже существует – есть открытые бесплатные библиотеки.

Библиотеки представляют собой уже написанные кем-то переносимые наборы проверенного кода. Это готовые решения, которые программисты могут присоединять к своим программам, вставлять их в свой код по специальным алгоритмам, причем в разных проектах.

Ошибки, обнаруженные компилятором, называются **ошибками времени компиляции**; ошибки, обнаруженные компоновщиком, называются **ошибками времени компоновки**, а ошибки, не обнаруженные на этих этапах и проявляющиеся при выполнении программы, называются **ошибками времени выполнения** или логическими ошибками. Как правило, ошибки времени компиляции легче понять и исправить, чем ошибки времени компоновки.

В свою очередь. ошибки времени компоновки легче обнаружить и исправить, чем ошибки времени выполнения.

Отладчик позволяет управлять процессом исполнения программы, является инструментом для поиска и исправления ошибок в программе.

Базовый набор функций отладчика включает:

- пошаговое выполнение программы (режим трассировки) с отображением результатов;
- остановка в заранее определенных точках;
- возможность остановки в некотором месте программы при выполнении некоторого условия;
- изображение и изменение значений переменных.

Загрузчик - системная обрабатывающая программа. Загрузчик помещает объектные и загрузочные модули в оперативную память, объединяет их в единую программу, корректирует перемещаемые адресные константы с учетом фактического адреса загрузки и передает управление в точку входа созданной программы.

Большинство программистов предпочитают использовать **интегрированную среду разработки** (Integrated Development Environment – IDE), объединяющую этапы программирования, компиляции и компоновки в пределах единого пользовательского

интерфейса, предоставляющего также средства отладки, облегчающие обнаружение ошибок и устранение проблем. Наиболее популярные IDE – Visual Studio, CLion, IntelliJ IDEA.