



История развития языков программирования

Язык программирования

Язык программирования - формальная знаковая система, предназначенная для записи программ.

Язык программирования - это набор символов (цифр, букв, специальных знаков) и система правил образования (синтаксис) и правил истолкования (семантика) конструкций из этих символов, с помощью которых описывается порядок выполнения алгоритма. Язык программирования имеет иерархическую структуру.

Обычно в нем выделяют четыре уровня:

- ✓ основные символы (алфавит);
- ✓ слова;
- ✓ выражения;
- ✓ предложения (операторы)

Со времени создания первых программируемых машин человечество придумало уже более **8,5 тыс. языков программирования**. Каждый год их число пополняется новыми.

Языки программирования высокого и низкого уровней

- Язык низкого уровня – программирование непосредственно в машинных кодах.
- Язык высокого уровня – (согласно ГОСТ 19781-90) язык программирования, понятия и структура которого удобны для восприятия человеком.

Языки программирования компилируемые и интерпретируемые

- Программа на компилируемом языке при помощи специальной программы-компилятора преобразуется (компилируется) в машинный код и далее записывается в исполняемый модуль, который может быть запущен на выполнение как отдельная программа.

Другими словами, компилятор переводит исходный текст программы с языка программирования высокого уровня в двоичные коды инструкций процессора.

- Если программа написана на интерпретируемом языке, то интерпретатор непосредственно выполняет построчно (интерпретирует) исходный текст без предварительного перевода. При этом программа остаётся на исходном языке и не может быть запущена без интерпретатора.

Этапы процесса компиляции

- **Лексический анализ.**

На этом этапе последовательность символов исходного файла преобразуется в последовательность лексем (лексема - допустимый символ или последовательность допустимых символов языка программирования, имеющая смысл для компилятора).

- **Синтаксический (грамматический) анализ.**

Последовательность лексем преобразуется в дерево разбора.

- **Семантический анализ.**

Дерево разбора обрабатывается с целью установления его семантики (смысла) - напр. привязка идентификаторов к их декларациям, типам, проверка совместимости, определение типов выражений и т. д. Результат обычно называется «промежуточным представлением/кодом»

- **Оптимизация.**

Выполняется удаление излишних конструкций и упрощение кода с сохранением его смысла.

- **Генерация кода.**

Из промежуточного представления выдается код на целевом языке.

Эволюция

языков программирования

Первые универсальные языки

Во времена первых компьютеров (40е гг. XXв.) программирование осуществлялось в машинных кодах, а носителями информации были перфокарты, перфоленты.

Ассемблер

Ассемблер (50-е гг. ХХв) – язык программирования низкого уровня.

В языке Ассемблер осуществлен переход к символическому кодированию машинных команд.

Появилась также возможность использования макросов и меток, что также упрощало создание, модификацию и отладку программ.

Фортран

В **1954** году в корпорации IBM группой разработчиков во главе с **Джоном Бэкусом** был создан язык программирования Fortran.

Это первый язык программирования высокого уровня.

Ключевой идеей, отличающей новый язык от ассемблера, была концепция подпрограмм.

Язык Фортран использовался (и используется по сей день) для научных вычислений.

Cobol

В 1960 году был создан язык программирования Cobol.

Он задумывался как язык для создания коммерческих приложений.

Отличительной особенностью языка является возможность эффективной работы с большими массивами данных, что характерно именно коммерческих приложений.

PL/1

В 1964 году корпорация IBM создала язык PL/1, который был призван заменить Cobol и Fortran в большинстве приложений

Язык обладал исключительным богатством синтаксических конструкций.

В нем впервые появилась обработка исключительных ситуаций и поддержка параллелизма

Из-за своей сложности язык не стал популярным

BASIC

В **1963** году в Дартмурском колледже был создан язык программирования **BASIC** (Beginners' All-Purpose Symbolic Instruction Code - многоцелевой язык символьных инструкций для начинающих).

Язык задумывался как средство обучения и как **первый изучаемый язык программирования.**

Он предполагался легко интерпретируемым и компилируемым.

Algol

В 1960 году командой во главе с Петером Науром (Peter Naur) был создан язык программирования Algol.

Этот язык дал начало целому семейству Алгол - подобных языков (важнейший представитель - Pascal).

Дальнейшее развитие языков программирования

Создание каждого из вышеупомянутых языков было вызвано некоторыми практическими требованиями.

Эти языки послужили фундаментом для более поздних разработок. Все они представляют одну и ту же парадигму программирования – **алгоритмическое программирование**.

Следующие языки пошли существенно дальше в своем развитии, в сторону более глубокого абстрагирования.

Сведения о более поздних языках будут приводиться в виде описания семейств языков.

Pascal-подобные языки

В 1970 году Никлаусом Виртом был создан язык программирования Pascal.

Язык замечателен тем, что это первый широко распространенный язык для структурного программирования.

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.

Основные правила структурного программирования

Любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:

- ▣ последовательное исполнение - однократное выполнение операций в том порядке, в котором они записаны в тексте программы;
- ▣ ветвление - однократное выполнение одной из двух или более операций, в зависимости от условия;
- ▣ цикл - многократное исполнение одной и той же операции до тех пор, пока выполняется условие продолжения цикла.

Повторяющиеся фрагменты программы и представляющие собой логически целостные вычислительные блоки могут оформляться в виде подпрограмм (процедур или функций).

Наиболее сильной критике со стороны разработчиков структурного подхода к программированию подвергся **оператор GOTO** (оператор безусловного перехода).

Структурное программирование – это программирование без GOTO.

Структурное программирование - программирование «сверху-вниз»

Программирование «сверху-вниз», нисходящее программирование предполагает последовательное разложение общей задачи на более мелкие простые подзадачи.

В результате строится иерархическая схема, отражающая состав и взаимоподчиненность отдельных задач, которая называется **«функциональная структура алгоритма (ФСА) приложения»**.

Дальнейшее развитие Pascal - подобных языков

Отрицательной чертой языка Pascal было отсутствие в нем средств для разбиения программы на модули.

Н.Вирт в дальнейшем разработал язык **Modula-2 (1978)**, в котором идея модуля стала одной из ключевых концепций языка. **В 1988 году появилась Modula-3**, в которую были добавлены объектно-ориентированные черты.

Логическим продолжением Pascal и Modula являются язык **Oberon** и **Oberon-2**. Они характеризуются движением в сторону объектно- и компонентно- ориентированности.

Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) - стиль программирования, в котором основными концепциями являются понятия **объектов** и **классов**.

Класс - это тип, описывающий устройство объектов - экземпляров. Класс можно сравнить с чертежом, согласно которому создаются объекты. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области.

Объект - это экземпляр класса.

Основные концепции ООП

- Система состоит из объектов;
- Объекты некоторым образом взаимодействуют между собой;
- Каждый объект характеризуется своим состоянием и поведением;
- Состояние объекта задаётся значением полей данных (свойствами);
- Поведение объекта задаётся методами.

Языки ООП

- **Языки** объектного программирования принято делить на **объектные**, в которых существуют классы и объекты, и **объектно-ориентированные**, в которых программист может не только пользоваться predetermined классами, но и задавать собственные пользовательские классы.
- **Объектно-ориентированное программирование** в настоящее время является абсолютным лидером в области прикладного программирования (языки Java, C#, C++, JavaScript, ActionScript и др.).

Основные понятия ООП

Наследование

- Наследованием называется возможность порождать один класс от другого с сохранением всех свойств и методов класса-предка (прародителя, иногда его называют суперклассом) и добавляя, при необходимости, новые свойства и методы. Набор классов, связанных отношением наследования, называют иерархией. Наследование призвано отобразить такое свойство реального мира, как иерархичность.

Полиморфизм

- Полиморфизмом называют явление, при котором один и тот же программный код (полиморфный код) выполняется по-разному в зависимости от того, объект какого класса используется при вызове данного кода.

Основные понятия ООП

Абстракция данных

- Объекты представляют собою упрощенное, идеализированное описание реальных сущностей предметной области. Если соответствующие модели адекватны решаемой задаче, то работать с ними оказывается намного удобнее, чем с низкоуровневым описанием всех возможных свойств и реакций объекта.

Инкапсуляция

- Инкапсуляция - это принцип, согласно которому любой класс должен рассматриваться как чёрный ящик - пользователь класса должен видеть и использовать только интерфейсную часть класса (т. е. список декларируемых свойств и методов класса) и не вникать в его внутреннюю реализацию.

Си-подобные языки

В 1972 году Керниганом и Ритчи был создан язык программирования **Си**. Он создавался как язык для разработки операционной системы **UNIX**.

Си позволяет эффективно работать с данными, предоставляя при этом структурированные управляющие конструкции и абстракции высокого уровня (структуры и массивы). Именно с этим связана его огромная популярность и поныне.

продолжение

В 1986 году Бьярн Страуструп создал первую версию языка **C++**, добавив в язык C объектно-ориентированные черты и исправив некоторые ошибки и неудачные решения языка.

C++ продолжает совершенствоваться и в настоящее время, так в 1998 году вышла новая (третья) версия стандарта, содержащая существенные изменения.

продолжение

В 1995 году в корпорации Sun Microsystems Кеном Арнольдом и Джеймсом Гослингом был создан язык **Java**. Он наследовал синтаксис С и С++.

Отличительной особенностью языка является компиляция в код некоей абстрактной машины, для которой затем пишется эмулятор (Java Virtual Machine) для реальных систем. Кроме того, в Java нет указателей и множественного наследования, что сильно повышает надежность программирования.

продолжение

В 1999–2000 годах в корпорации Microsoft был создан язык C#.

Он в достаточной степени схож с Java (и задумывался как альтернатива последнему), но имеет и отличительные особенности.

Ориентирован, в основном, на разработку многокомпонентных Интернет-приложений.

Языки Ada и Ada 95

В 1983 году под эгидой Министерства Обороны США был создан язык Ada.

В 1995 году был принят стандарт языка Ada 95, который развивает предыдущую версию, добавляя в нее объектно - ориентированность и исправляя некоторые неточности.

Оба этих языка не получили широкого распространения вне военных и прочих крупномасштабных проектов (авиация, железнодорожные перевозки).

Основной причиной является сложность освоения языка и достаточно громоздкий синтаксис (значительно более громоздкий, чем Pascal).

Языки обработки данных

Все вышеперечисленные языки являются языками общего назначения в том смысле, что они не ориентированы и не оптимизированы под использование каких-либо специфических структур данных или на применение в каких-либо специфических областях.

Было разработано большое количество языков, ориентированных на достаточно **специфические применения**. Ниже приведен краткий обзор таких языков

APL

В 1957 году была предпринята попытка создания языка для описания математической обработки данных.

Язык был назван APL (Application Programming Language).

Его отличительной особенностью было использование математических символов (что затрудняло применение на текстовых терминалах; появление графических интерфейсов сняло эту проблему) и очень мощный синтаксис

Snobol и Icon

В 1962 году появился язык Snobol (а в 1974 - его преемник Icon), предназначенный для обработки строк.

Синтаксис Icon напоминает C и Pascal одновременно. Отличие заключается в наличии мощных встроенных функций работы со строками и связанная с этими функциями особая семантика.

Современным аналогом Icon и Snobol является Perl - язык обработки строк и текстов, в который добавлены некоторые объектно-ориентированные возможности.

SETL

В 1969 году был создан язык SETL - язык для описания операций над множествами.

Основной структурой данных в языке является множество, а операции аналогичны математическим операциям над множествами.

Lisp и ему подобные языки

В 1958 году появился язык Lisp - язык для обработки списков.

Получил достаточно широкое распространение в системах искусственного интеллекта.

Имеет несколько потомков: Planner (1967), Scheme (1975), Common Lisp (1984).

Многие его черты были унаследованы современными языками функционального программирования.

Скриптовые языки

В последнее время в связи развитием Интернет-технологий, широким распространением высокопроизводительных компьютеров и рядом других факторов получили распространение так называемые скриптовые языки.

Характерными особенностями данных языков являются:

во-первых, их интерпретируемость (компиляция либо невозможна, либо нежелательна),
во-вторых, простота синтаксиса,
в-третьих, легкая расширяемость.

JavaScript

Язык был создан в компании Netscape Communications в качестве языка для описания сложного поведения веб-страниц. Интерпретируется браузером во время отображения веб-страницы.

По синтаксису схож с Java и отдаленно с C/C++.

VBScript

Язык был создан в корпорации Microsoft во многом в качестве альтернативы JavaScript.

Имеет схожую область применения.

Синтаксически схож с языком Visual Basic (и является усеченной версией последнего).

Так же, как и JavaScript, исполняется браузером при отображении веб-страниц и имеет ту же степень объектно-ориентированности

Perl

Язык создавался в помощь системному администратору операционной системы Unix для обработки различного рода текстов и выделения нужной информации.

Развился до мощного средства работы с текстами

Является интерпретируемым языком и реализован практически на всех существующих платформах.

Применяется при обработке текстов, а также для динамической генерации веб-страниц на веб-серверах.

Python

Интерпретируемый объектно-ориентированный язык программирования.

По структуре и области применения близок к Perl, однако менее распространен и более строг и логичен.

Имеются реализации для большинства существующих платформ

Объектно-ориентированные ЯЗЫКИ

Объектно-ориентированный подход, пришедший на смену структурному, впервые появился отнюдь не в С++, как полагают некоторые.

Существует целая череда чистых объектно-ориентированных языков

Simula

Первым объектно-ориентированным языком был язык Simula (1967).

Этот язык был предназначен для моделирования различных объектов и процессов, и объектно-ориентированные черты появились в нем именно для описания свойств модельных объектов.

Smalltalk

Популярность объектно-ориентированному программированию принес язык Smalltalk, созданный в 1972 году.

Язык предназначался для проектирования сложных графических интерфейсов и был первым по-настоящему объектно-ориентированным языком.

В нем классы и объекты - это единственные конструкции программирования

Языки параллельного программирования

Большинство компьютерных архитектур и языков программирования ориентированы на последовательное выполнение операторов программы.

В настоящее время, однако же, существуют программно-аппаратные комплексы, позволяющие организовать параллельное выполнение различных частей одного и того же вычислительного процесса.

Язык Occam (1982) и модель параллельных вычислений Linda (1985)

Неимперативные языки

Программы на императивных языках представляют собой пошаговое описание решения той или иной задачи.

Можно попытаться описывать лишь постановку проблемы, а решать задачу поручить компилятору.

Существует два основных подхода, развивающие эту идею: **функциональное** и **логическое** программирование

Функциональные языки

Основная идея, лежащая в основе функционального программирования, - это **представление программы в виде математических функций** (т.е. функций, значение которых определяется лишь их аргументами, а не контекстом выполнения).

Из языков с энергичной семантикой упомянем **ML** и два его современных диалекта - **Standard ML (SML)** и **CaML**.

Последний имеет объектно-ориентированного потомка - **Objective CaML (O'CaML)**.

Среди языков с ленивой семантикой наиболее распространены два: **Haskell** и его более простой диалект **Clean**.

Логические языки

Программы на языках логического программирования выражены как формулы математической логики, а компилятор пытается получить следствия из них.

Родоначальником большинства языков логического программирования является язык **Prolog** (1971).

У него есть ряд потомков - Parlog (1983, ориентирован на параллельные вычисления), Delta Prolog и др.