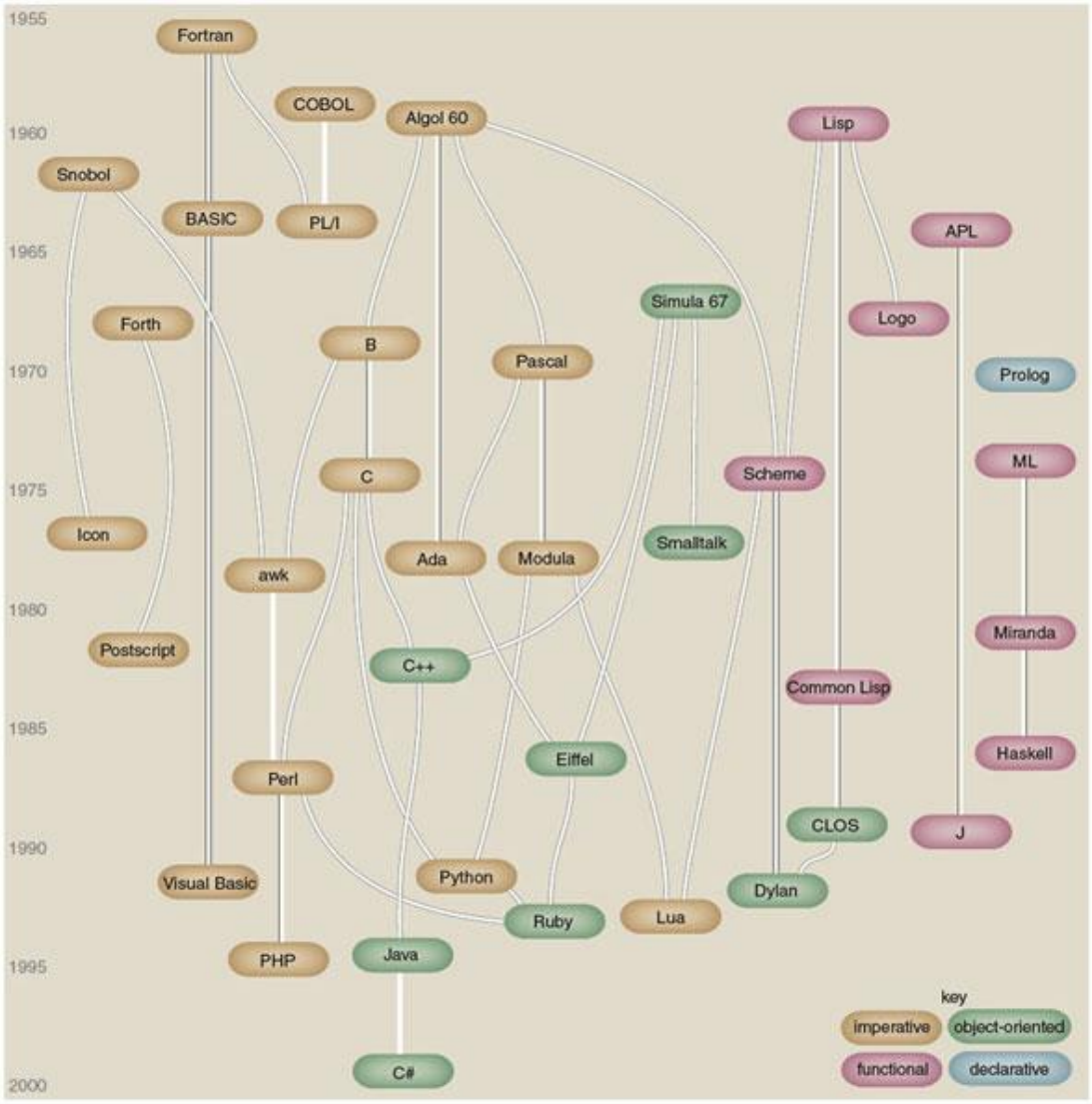
Fluxo Temporal das Principais Linguagens



Propósito:

Cronologia para o desenvolvimento das linguagens de programação, explorando o contexto em que elas foram desenvolvidas e a motivação para seu desenvolvimento.

Paradigmas de Linguagens

Um paradigma é o que determina o ponto de vista da realidade e como se atua sobre ela, os quais são classificados quanto ao seu conceito de base, podendo ser: Imperativo, funcional, lógico, orientado a objetos e estruturado. Cada qual determina uma forma particular de abordar os problemas e de formular respectivas soluções. Além disso, uma linguagem de programação pode combinar dois ou mais paradigmas para potencializar as análises e soluções.

Paradigma Imperativo

O Paradigma Imperativo é baseado na arquitetura de Von Neumann. É o primeiro paradigma a existir e até hoje é o dominante. Esse paradigma segue o conceito de um estado e de ações que manipulam esse estado, nele encontramos procedimentos que servem de mecanismos de estruturação.

É possível denominá-lo de procedural por incluir subrotinas ou procedimentos para estruturação.

Exemplos de linguagens de programação que se baseiam no modelo imperativo:

- Ada;
- ALGOL;
- Assembler;
- Basic;
- C;
- COBOL;
- Fortran;
- Pascal;
- Python;
- Lua.

Vantagens:

- As vantagens desse paradigma são: eficiência (porque embute o modelo de Von Neumann); modelagem "natural" de aplicações do mundo real; paradigma dominante e bem estabelecido; e também muito flexível.

Desvantagens:

- Difícil legibilidade; as instruções são centradas no como e não no o que.

Paradigma Estruturado

O paradigma preconiza que todos os programas possíveis podem ser reduzidos a apenas três estruturas: sequência, decisão e iteração. Tendo, na prática, sido transformada na Programação modular, a Programação estruturada orienta os programadores para a criação de estruturas simples nos programas, usando as subrotinas e as funções. Foi a forma dominante na criação de software entre a programação linear e a programação orientada por objetos.

Apesar de ter sido sucedida pela programação orientada por objetos, pode-se dizer que a programação estruturada ainda é marcantemente influente, uma vez que grande parte das pessoas ainda aprende programação através dela.

Linguagens

- C;
- Basic;
- Pascal;
- COBOL.

Vantagens:

- Os problemas podem ser quebrados em vários subproblemas, a boa legibilidade e a boa compreensão da estrutura deste paradigma motivam os programadores a iniciarem a programação pelo modelo estruturado.

Desvantagens:

- Os dados são separados das funções; Mudança na estrutura dos dados acarreta alteração em todas as funções relacionadas; Gera sistemas difíceis de serem mantidos.

Paradigma Orientado a Objetos

A programação Orientada a Objetos é baseada na composição e interação de diversas unidades de softwares denominados objetos. O funcionamento de um software orientado a objetos se dá através do relacionamento e troca de mensagens entre esses objetos. Esses objetos são classes, e nessas classes os comportamentos são chamados de métodos e os

estados possíveis da classe são chamados de atributos. Nos métodos e nos atributos também são definidas as formas de relacionamento com outros objetos.

Linguagens

Exemplos de linguagens de programação que se baseiam no modelo orientado a objetos:

- Smalltalk;
- Python;
- Ruby;
- C++;
- Object Pascal;
- Java;
- C#;
- Oberon;
- Ada;
- Eiffel;
- Simula;
- .NET.

Vantagens: Esse paradigma possui todas as vantagens do paradigma imperativo entre outras:

- A alteração de um módulo não incorre na modificação de outros módulos;
- Quanto mais um módulo for independente, maior a chance de ele poder ser reutilizado em outra aplicação.

Desvantagens:

- Por exigir formas de pensar relativamente complexas, a programação orientada a objetos até hoje ainda não é bem compreendida ou usada pela maioria.

Paradigma Funcional

Este paradigma trata a computação como uma avaliação de funções matemáticas. Este método enfatiza a aplicação de funções, as quais são tratadas como valores de primeira importância, ou seja, funções podem ser parâmetros ou valores de entrada para outras funções e podem ser os valores de retorno ou saída de uma função.

Linguagens

Exemplos de linguagens de programação que se baseiam no modelo funcional:

- Lambda (não implementado para computadores);
- LISP;
- Scheme (tentativa de simplificar e melhorar o LISP);
- ML (Criada em universidade);
- Miranda (também criada em universidade);
- Haskell;

Desvantagens:

- Na programação funcional parecem faltar diversas construções freqüentemente (embora incorretamente) consideradas essenciais em linguagens imperativas, como C. Por exemplo, não há alocação explícita de memória nem de variáveis.

Vantagens:

- Devido ao processo automático de alocação de memória, então efeitos colaterais no cálculo da função são eliminados. Sem estes efeitos, a linguagem assegura que o resultado da função será o mesmo para um dado conjunto de parâmetros não importando onde, ou quando, seja avaliada e é empregado em computações independentes para execução paralela. A recursividade em programação funcional pode assumir várias formas e é em geral uma técnica mais poderosa que o uso de laços do paradigma imperativo.

Paradigma Lógico

Nesse paradigma programas são relações entre Entrada/Saída. Possui estilo declarativo, como o paradigma funcional. Inclui características imperativas, por questões de eficiência. Aplicações em prototipação em geral, sistemas especialistas, bancos de dados, etc.

Linguagens

Exemplos de linguagens de programação que se baseiam no paradigma lógico:

- Popler;
- Conniver;
- QLISP;
- Planner;
- Prolog;
- Mercury;
- Oz;
- Frill.

Vantagens:

- Possui a princípio todas as vantagens do paradigma funcional. E permite concepção da aplicação em um alto nível de abstração (através de associações entre E/S).

Desvantagens:

- Variáveis de programa não possuem tipos, nem são de alta ordem.

Linha Temporal das Linguagens

PLANKALKÜL 1945 — Criada por Konrad Zuse

- Foi a primeira linguagem de programação de alto nível do mundo
- Nunca foi implementada;
- Estruturas de dados avançadas:
  - reais, arrays, records.

FORTRAN 1954 — The IBM Mathematical FORmula TRANslating System: FORTRAN por John Backus

- Projetado para o IBM 704
- Ambiente de desenvolvimento:
  - Computadores tinham pouca capacidade de processamento e memória e não eram confiáveis Aplicações científicas Sem metodologia ou ferramentas de programação

FORTRAN I 1957

- Primeira versão implementada do FORTRAN
- I/ O formatado
- Nomes podiam ter até 6 caracteres
- Subprogramas definidos pelo usuário
- Comando de seleção de t rês modos (IF aritmético)
- Loop de contagem pós-testado (DO)
- Compilador lançado em abril de 1957, depois de 18 homens-ano de esforço.
- Programas maiores que 400 linhas raramente compilavam corretamente, principalmente devido à baixa confiabilidade do 704 Código era muito rápido Rapidamente tornou- se popular

FORTRAN II 1958

- Corrigiu vários “bugs” do Fortran I Compilação de subrotinas independente 50% do código escrito para o IBM 704 era FORTRA

FORTRAN IV - 1960-62

- FORTRAN III nunca largamente distribuído
- Declaração de tipos explícita
- Comando de seleção lógica
- Nomes de subprogramas podiam ser parâmetros
- Padrão ANSI em 1966

FORTRAN 77
1978

- Manuseio de *character; string*
- Comando de controle de *loop* lógico
- Comando IF- THEN- ELSE

FORTRAN 90
1990

- Funções built-in para operações com arrays
- Arrays dinâmicos
- Ponteiros
- Tipo registro
- Recursão
- Novas estruturas de controle, ex. CASE
- Checagem de tipo de parâmetro
- Módulos

LISP
1959

- Somente dois tipos de dados: átomos e listas
- Sintaxe baseada no *lambda calculus*
- Pioneira da programação funcional
- Não necessita de variáveis ou atribuição
- Controle via recursão e expressões condicionais
- Paradigma alternativo ao modelo imperativo
- Ainda uma das linguagens dominantes

ALGOL 58
1958

- Conceito de tipo formalizado
- Nomes podiam ter qualquer tamanho
- Arrays podiam ter um número qualquer de índices
- Parâmetros eram separados por modo (entrada/ saída)
- Índices eram colocados entre colchetes
- Comandos de composição de blocos (begin ...end)
- Ponto e vírgula como separador de comandos
- Operador de atribuição **:=**
- IF com cláusula ELSE- IF

ALGOL 60
1960

- Modificou o ALGOL 58 durante um encontro de 6 dias em Paris:
  - Estrutura de blocos (escopo local)
  - Dois modos de passagem de parâmetros
  - Subprogramas recursivos
  - Arrays em pilhas dinâmicas
  - Ainda nenhum I/ O nem manuseio de strings
- Sucessos:
  - Foi o padrão para publicação de algoritmos por mais de 20 anos
  - Todas as linguagens imperativas são baseadas nela
  - Primeira linguagem independente de máquina
  - Primeira linguagem cuja sintaxe foi definida formalmente (BNF)
- Falha:
  - Nunca foi amplamente utilizada, especialmente nos EUA
- Motivos:
  - Nenhum I/ O e o conjunto de caracteres faziam com que os programas não fossem muito portáveis
  - Muito flexível, portanto difícil de implementar Falta de apoio da IBM

ALGOL 68
1968

- A partir do desenvolvimento do ALGOL 60, mas não é um superconjunto daquela linguagem
- Projeto é baseado no conceito de ortogonalidade Fonte de várias idéias adotadas por outras linguagens
- Contribuições:
  - Estruturas de dados definidas pelo usuário Arrays dinâmicos (chamados flex arrays)

SIMULA 67
1967

- Baseada no SIMULA I, criada por Kristen Nygaard e OleJohan Dahal, entre 1962 e 1964 na Noruega
- Projetada inicialmente para simulação de sistemas
- Baseada no ALGOL 60 e SIMULA I
- Contribuições:
  - Co-rotinas: um tipo de subprograma que pode ser reiniciado a partir do ponto onde previamente havia parado POO

COBOL
1960

- Objetivos do projeto:
- Deve parecer com inglês simples
- Deve ser fácil de usar, mesmo que signifique menos recursos
- Deve ampliar a base de usuários de computadores
- Não deve ser influenciado por problemas de implementação (compiladores)
- Contribuições:
  - Estruturas de dados hierárquicas (registros)
  - Comandos de seleção aninhados
  - Nomes longos (até 30 caracteres), com hífens Data Division

PL/ I
1965

Projetado pela IBM e SHARE

- Incluía o que era considerado o melhor do: ALGOL 60:
- Recursão
- Estrutura de blocos
- **FORTRAN IV**
  - Compilação de rotinas em separado
  - Comunicação por dados globais
- **COBOL 60**
- Estruturas de dados
  - I/ O (facilidades para geração de relatórios)
  - Linguagem grande e complexa
- Utilizada em aplicações científicas e comerciais
- Sucesso parcial
- Contribuições:
  - Primeiro tratamento de exceção (23 tipos)
  - Tipo de dado ponteiro
  - Referência a seções de arrays

Pascal
1971
Niklaus Wirth

- Baseada no ALGOL W, proposta alternativa apresentada pelos que consideraram a ALGOL68 muito complexa
- Simplicidade e tamanho reduzido eram objetivos de projeto
- Projetada para o ensino de programação estruturada
- Ainda é a uma das LPs mais usadas para o ensino de programação estruturada nas universidades

BASIC
1964
Kemeny & Kurtz em Dartmouth

Objetivos de Projeto:

- Fácil de aprender e usar (por outros que não fossem estudantes de ciências)
- Linguagem “agradável e amigável”
- Rápida implementação de pequenas aplicações
- Tempo do usuário mais importante que tempo de máquina
- Versão inicial era compilada, sem entrada pelo terminal
- Simples: 14 tipos de comandos e 1 tipo de dados

Dialetos populares atuais: QuickBASIC e Visual BASIC

Modula2
1975-76
Niklaus Wirth

- Recursos do Pascal mais módulos, suportando tipos abstratos de dados, procedimentos como tipos e características de baixo nível projetadas para programação de sistemas.

Modula3
1988-89
Digital e Olivetti Research Center

- Baseada no Modula2, Mesa, Cedar e Modula2+;
- Adiciona ao Modula2:
- POO;
- Tratamento de exceção;
- Garbage collection;
- Concorrência.

Delphi Borland
1995

- Pascal mais características de suporte a POO
- Mais elegante e seguro que C+ +
- Baseada em Object Pascal, projetada anos antes.

C
1972

Lab. Bell por Dennis Richie

- Ancestrais incluem CPL, BCPL, B e ALGOL68
- Operadores poderosos, checagem de tipos pobre
- Inicialmente popular entre usuários UNIX

Prolog
1972

Desenvolvido na Universidade de AixMarseille, por Comerauer e Roussel, com ajuda de Kowalski da Universidade de Edinburgh

- Baseada em lógica de predicados de 1a.ordem Não procedural
- LP declarativa que usa um padrão de inferência (resolução por refutação) para inferir se expressões fornecidas são verdadeiras (e sob que condições) ou não.

Ada
1983

Contribuições:

- Suporte para abstração de dados
- Facilidades para tratamento de exceções
- Unidades de programa genéricas: versões instanciadas geradas pelo compilador de acordo com tipos de dados específicos
- Suporte à execução concorrente de unidades de programa

Comentários:

- Projeto via contrato competitivo
- Incluiu tudo que se conhecia sobre engenharia de software e projeto de linguagens
- Não confiável por causa da alta complexidade
- Compiladores de difícil construção.
- Primeiro compilador operacional surgiu 4 anos depois que o projeto da linguagem tinha sido completado.

Ada 95
1988

- Facilidades para interface gráfica
- Suporte para POO
- Mecanismos de controle para dados compartilhados
- Apesar de tão completa, Ada só foi largamente utilizada pela organização que promoveu seu desenvolvimento: o DoD do governo americano.

Smalltalk
1972- 1980

Desenvolvida: Xerox PARC, Alan Kay, depois por Adele Goldberg

- Primeira implementação completa de uma linguagem orientada por objetos (abstração de dados, herança e vinculação de tipos dinâmica)
- Pioneira na interface gráfica baseada em janelas.

C+ +
1985

Desenvolvida nos Lab. Bell por Stroustrup Baseada no C and SIMULA 67

- Facilidades para POO do SIMULA 67 adicionadas ao C
- Possui tratamento de exceção
- Linguagem grande e complexa, em parte por suportar programação procedural e orientada por objetos
- Popularizou-se rapidamente, junto com POO

Java
1995

Desenvolvida pela Sun no começo dos anos de 1990

- Para POO, baseada em C++, mas significativamente simplificada (não usa ponteiros)
- Inclui suporte a applets e concorrência
- Padrão ANSI aprovado em novembro de 1997

Lua **1993**	Criada por Roberto Ierusalimschy, Luiz Henrique de Figueiredo e Waldemar Celes, membros da Computer Graphics Technology Group na PUC-Rio, a Pontifícia Universidade Católica do Rio de Janeiro, no Brasil. Versões da Lua antes da versão 5.0 foram liberadas sob uma licença similar à licença BSD. A partir da versão 5.0, Lua foi licenciada sob a licença MIT. **Lua** é uma **linguagem** de script de multiparadigma, pequena, reflexiva e leve, projetada para expandir aplicações em geral, por ser uma **linguagem** extensível (que une partes de um programa feitas em mais de uma **linguagem**), para prototipagem e para ser embarcada em softwares complexos, como jogos.

Wolfram Language **2013**	É uma linguagem de programação multi-paradigma desenvolvida pela Wolfram Research, é a linguagem de programação de Mathematica e a programação Wolfram Cloud. Ele enfatiza a computação simbólica, a programação funcional e a programação baseada em regras e pode empregar estruturas e dados arbitrários. Ele inclui funções internas para gerar e executar máquinas de Turing, criar gráficos e áudio, analisar modelos 3D, manipulações de matrizes e resolver equações diferenciais. É extensivamente documentado.

Outras Linguagens

- ML (1973) – Meta Language. Linguagem de programação funcional. Versão mais recente é o SML/ NJ.
- Clipper (1984) – Linguagem de desenvolvimento de banco de dados para DOS.
- SQL – Structured Query Language. Padrão para operações em bancos de dados relacionais.
- Perl (1987) – Linguagem script com facilidades para manipulação de texto.
- PYTHON (1991) – Linguagem interpretada, orientada a objetos similar ao Perl.
- PHP (1995) - Linguagem script para o desenvolvimento de aplicações Web; embutível dentro do HTML.
- ASP – Active Server Page. Linguagem script para construção de páginas dinâmicas.
- JavaScript (1995) – Linguagem script para aplicações web executadas no cliente (CSI).
- HTML – Hipertext Markup Language, linguagem de marcação para formatação de hipertextos.
- XML – Extensible Markup Language. Usado para estruturação do conteúdo de páginas HTML.
- TeX/ LaTeX (1985) – Linguagem para formatação de textos científicos.
- PostScript (1982), PDF – Linguagens de formatação de textos para visualização.