# INE5416
# Paradigmas de Programação

Ricardo Azambuja Silveira
INE-CTC-UFSC
E-Mail: silveira@inf.ufsc.br
URL: www.inf.ufsc.br/~silveira

# Porque estudar LPs

- Capacidade de expressar idéias
- Capacidade de escolha de linguagens apropriadas
- Capacidade para aprender novas linguagens
- Entender melhor a importância da implementação
- Capacidade de projetar novas linguagens
- Avanço geral da computação

# Domínios de programação

- **Aplicações cientificas**
- **Aplicações comerciais**
- **Inteligência Artificial**
- **Programação de sistemas**
- **Linguagens de *scripting***
- **Linguagens de propósitos especiais**

# Critérios de avaliação de uma linguagem

- Legibilidade
  - Fatores:
    - Simplicidade global
    - Ortogonalidade
    - Instruções de controle
    - Tipos de dados e estruturas
    - Sintaxe
      - Formas identificadoras
      - Palavras especiais
      - Forma e significado

# Critérios de avaliação de uma linguagem

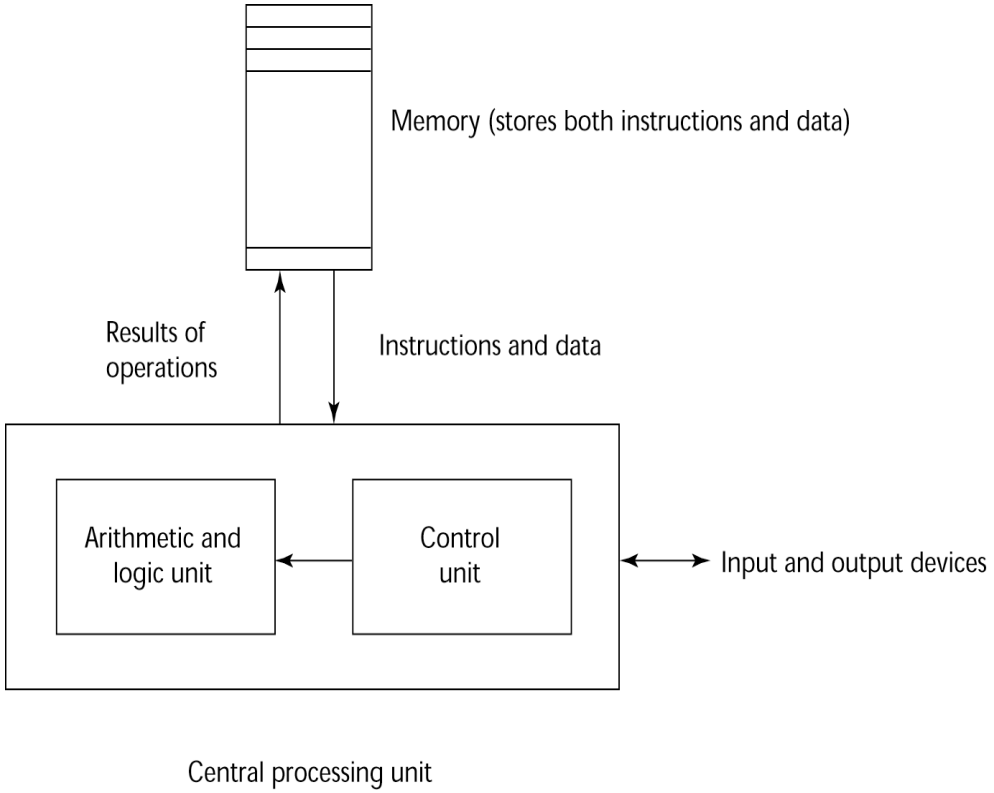
- Capacidade de escrita
  - Fatores:
    - Simplicidade e ortogonalidade
    - Suporte para abstração
    - Expressividade
- Confiabilidade
  - Fatores:
    - Verificação de tipos
    - Manipulaçào de excessões
    - Apelidos
    - Legibilidade e capacidade de escrita

# Critérios de avaliação de uma linguagem

- Custos
  - Categorias:
    - Treinamento dos programadores
    - Criação do software
    - Compilação
    - Execução
    - Ferramentas de desenvolvimento
    - Confiabilidade
    - Manutenção
- Outros
  - Portabilidade
  - Generalidade
  - Boa definição (padronização)

# Influências sobre o projeto

- Arquitetura do computador
  - A predominância das linguagens imperativas é influenciada pela arquitetura de von Neumann



# Influências sobre o projeto

- **Metodologias de programação**
  - **década de 50 e começo dos anos 60: simples aplicações;**
    - **preocupação com a eficiência das máquinas**
  - **Final dos anos 60:**
    - **A eficiência das pessoas se torna importante;**
      - **preocupação com a legibilidade, melhores estruturas de controle**
  - **Final dos anos 70:**
    - **abstração de dados**
  - **Meados dos anos 80*:***
    - **Programação orientada a objetos**

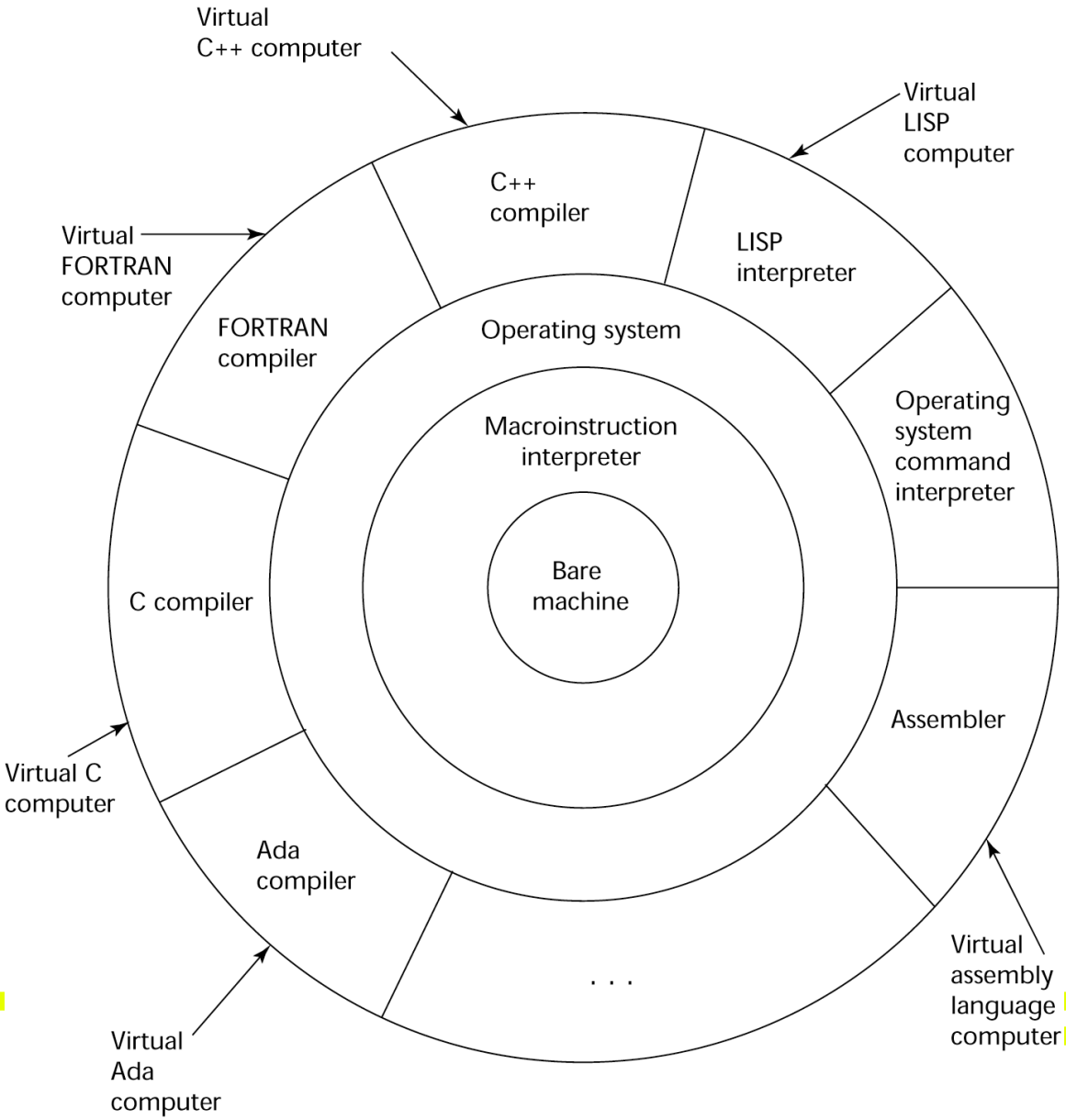
# Categorias de linguagens

- Imperativas
- Funcionais
- Lógicas
- Orientadas a objeto

# Relação custo benefício

1. **Confiabilidade versus custo de execução**
2. **Facilidade de escrita versus legibilidade**
3. **Flexibilidade versus segurança**

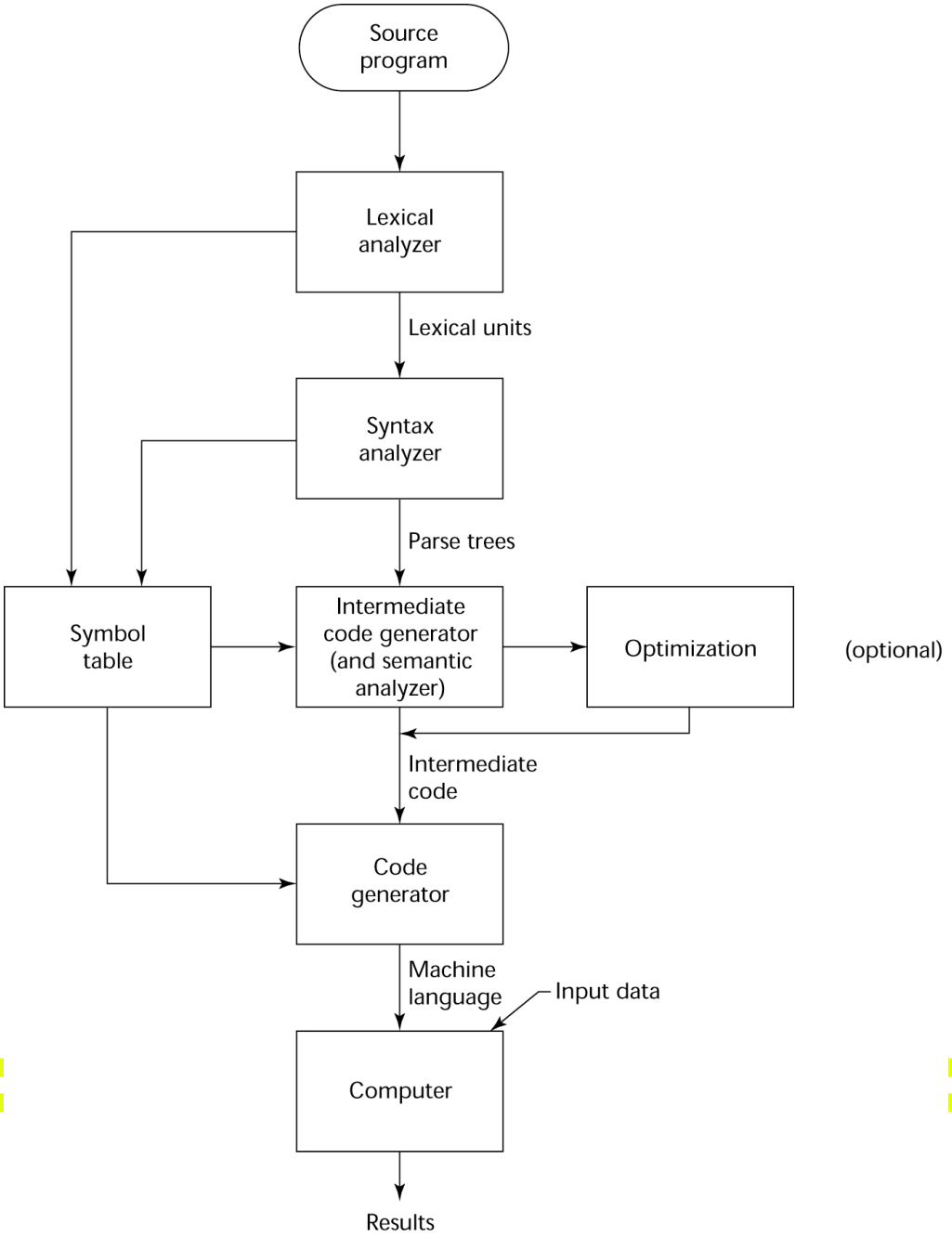
# Métodos de implementação



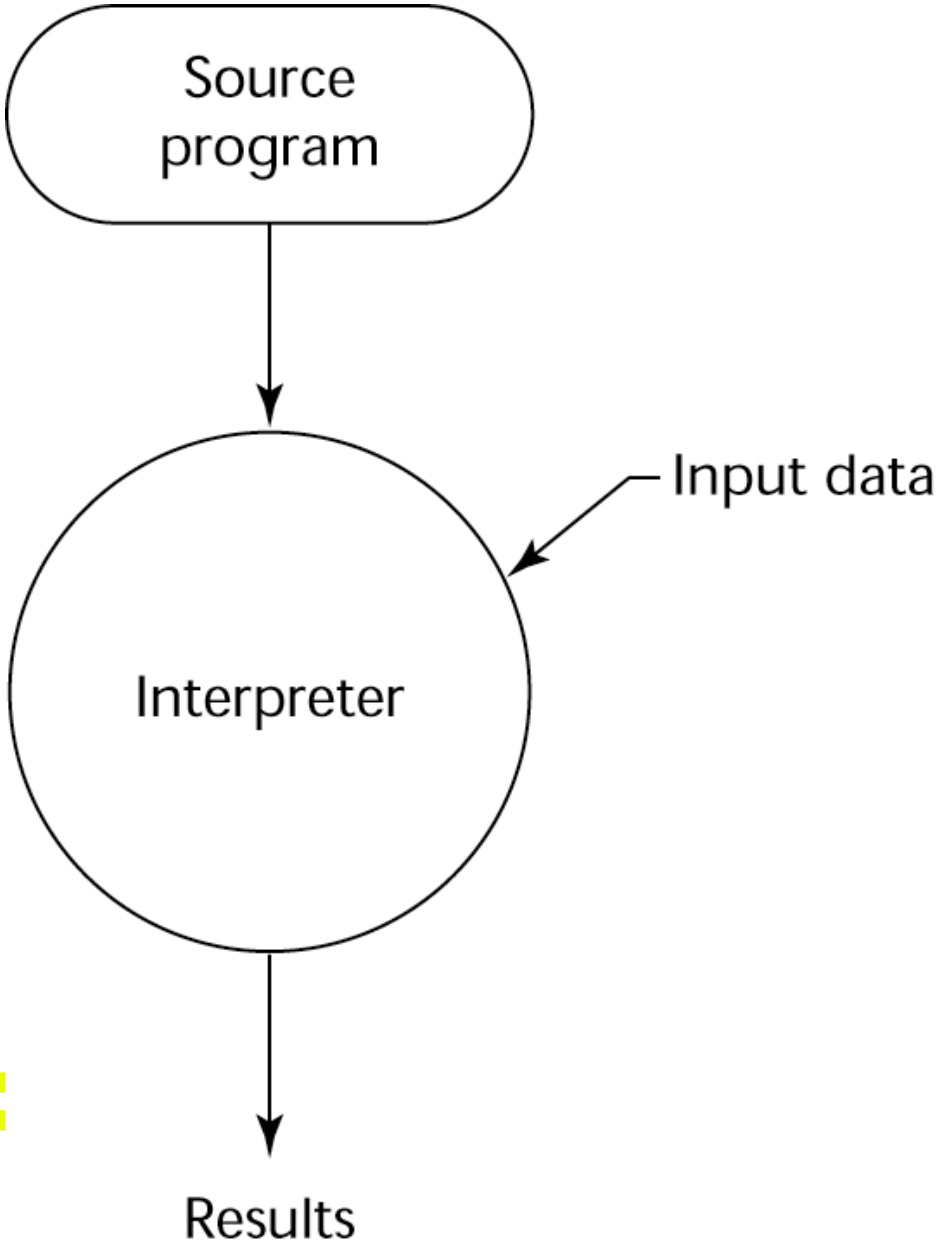
# Métodos de implementação

- Compilação
  - Traduz um programa escrito em linguagem de alto nível em um código de máquina
  - Tradução demorada
  - Execução mais rápida
- Intepretação pura
  - Sem tradução do código
  - Execução lenta
- Sistemas híbridos
  - Baixo custo de tradução
  - Velocidade de execução média

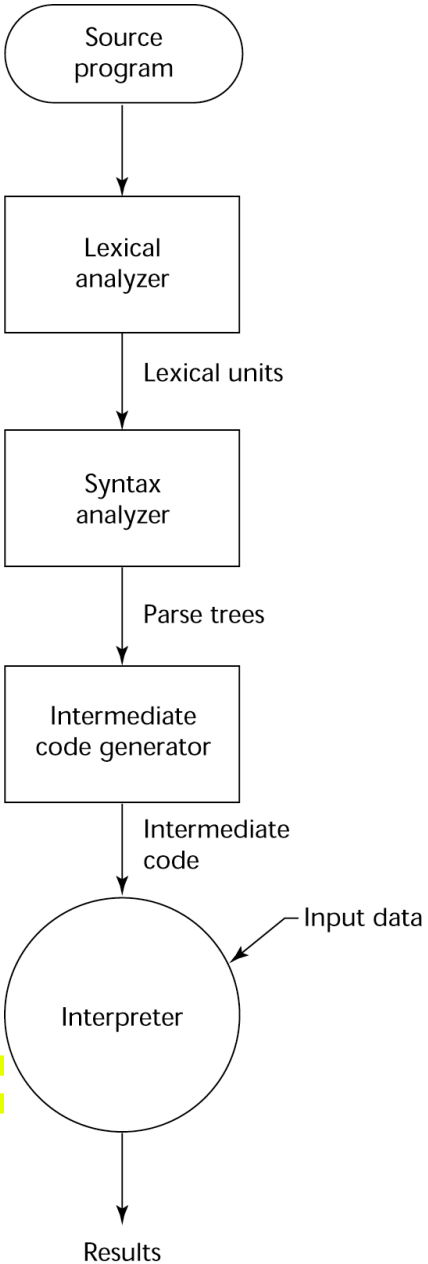
# Compilação


Source program
Lexical analyzer
Lexical units
Syntax analyzer
Parse trees
Symbol table
Intermediate code generator (and semantic analyzer)
Optimization
(optional)
Intermediate code
Code generator
Machine language
Input data
Computer
Results

# Interpretação pura



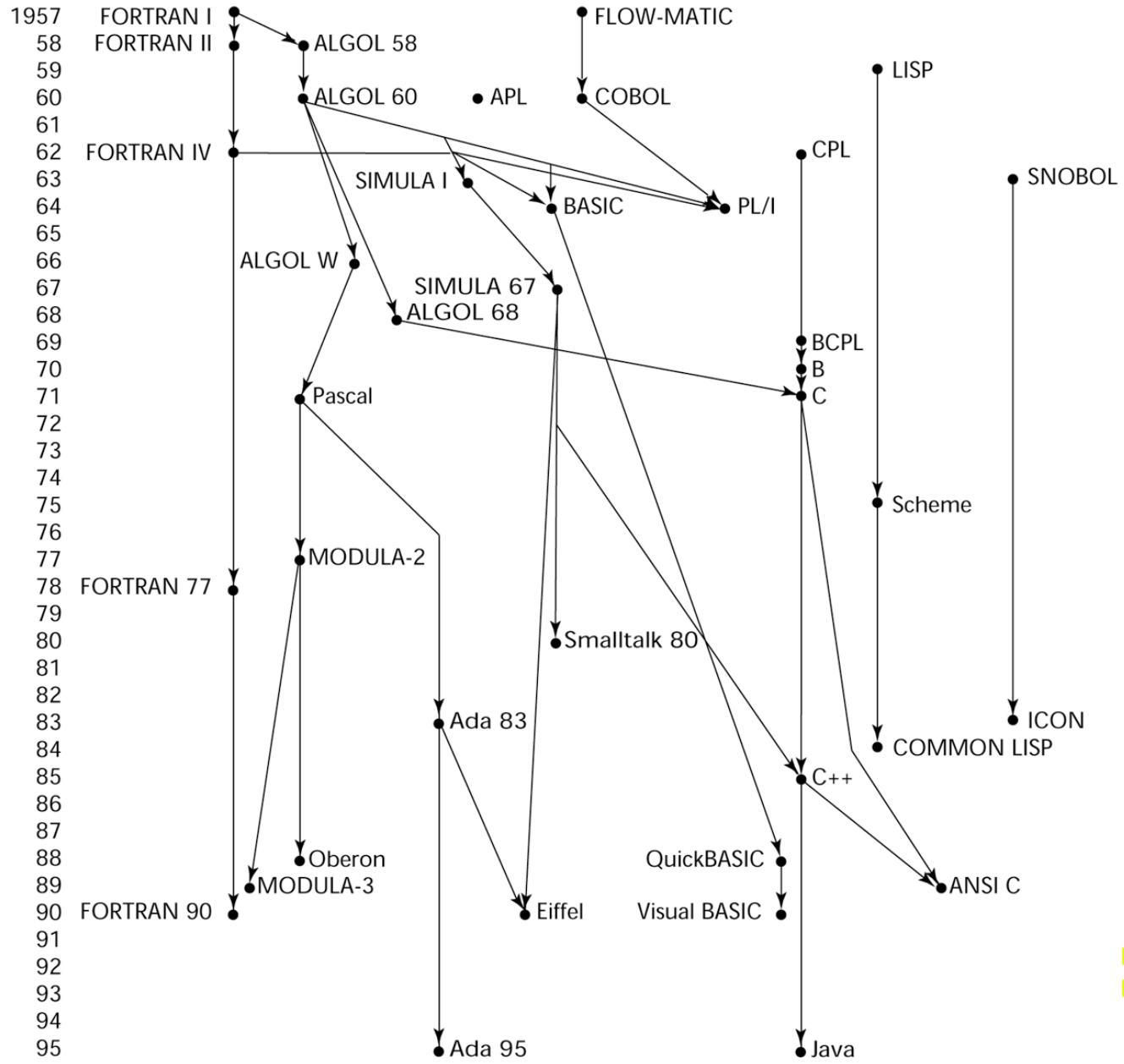
# Sistemas híbridos



# Ambientes de programação

- Conjunto de ferramentas usadas no desenvolvimento de  software
- Exemplos
  - UNIX
    - Um velho sistema operacional e um conjunto de ferramentas
  - Borland C++
    - Um ambiente para PC para  C and C++
  - Smalltalk
    - Um ambiente e processador de linguagem
  - Visual C++, Delphi, JBuilder, Visual Basic, etc
    - Grandes e complexos ambientes de programação visual

# Genealogia das linguagens



# Plankalkül

- **Produzida entre 1943 e 1945 por Konrad Zuse**
- **Nunca implementada**
- **Avançada estrutura de dados:**
  - **floating point, arrays, records**
  - **Invariants**
- **Notação:**

**A(7) := 5 * B(6)**

```
    |  5  *  B  =>  A
V   |        6      7          (subscripts)
S   |       1.n    1.n         (data types)
```

# Pseudocódigos

- Porque não usar código de máquina
  - péssima legibilidade
  - difícil de modificar
  - processo de codificação tedioso
  - Deficiências de hardware
    - endereços absolutos

*Short code*

*Desenvolvida em  1949 para o computador BINAC por John Mauchly*

- Versões codificadas de expressões matemáticas representadas por 72 bits agrupadas em 12 bytes de de 6 bits

# Pseudocódigos

- Speedcoding; 1954; IBM 701, John Backus
  - Pseudo-instruções para operações aritméticas e funções
  - Desvios condicional e incondicional
  - Auto incremento de registros de endereços
  - Lento! Instrução ADD demorava 4,2 ms
  - Somente 700 palavras livres para o programa do usuário
- Compilador UNIVAC: Grace Hopper, entre 1951 e 1953
- David Wheeler, 1950: endereços realocáveis
- Maurice Wilkes, 1951: Sub-rotinas

# Fortran I - 1957

- Baseado no Fortran 0 – 1954 (não implementado)
- Desenhado para o IBM 704
  - Registrador de endereços
  - Ponto flutuante em hardware
- Ambiente de desenvolvimento:
  - Máquinas pequenas e não confiáveis
  - Aplicações científicas
  - Não havia metodologia de programação nem ferramentas
  - Eficiência da máquina era o mais importante
- Impacto do ambiente no projeto
  - Nenhuma necessidade de armazenamento dinâmico
  - Necessidade de mecanismos eficientes de manuseio de matrizes e contadores de laço
  - Nenhum suporte a manuseio de string, aritmética decimal ou manipulação de entrada e saída (necessidades comerciais)

# Fortran I

- Primeira implementação comercial:
  - Nomes com até 6 caracteres
  - Laços com contador pós-teste (do while)
  - Entrada e saída formatada
  - Subprogramas definidos pelo usuário
  - Comandos de seleção em três modos (IF aritmético)
  - Nenhum comando de tipo
  - Sem compilação em separado
  - Compilador realizado em abril de 1957 depois de 18 homens-ano de trabalho
  - Programas maiores que 400 linha raramente compilavam
  - Código rápido
  - Tornou-se largamente utilizado
- FORTRAN II (1958):
  - Compilação independente de sub-rotinas
  - Correção de erros

# Evolução do Fortran

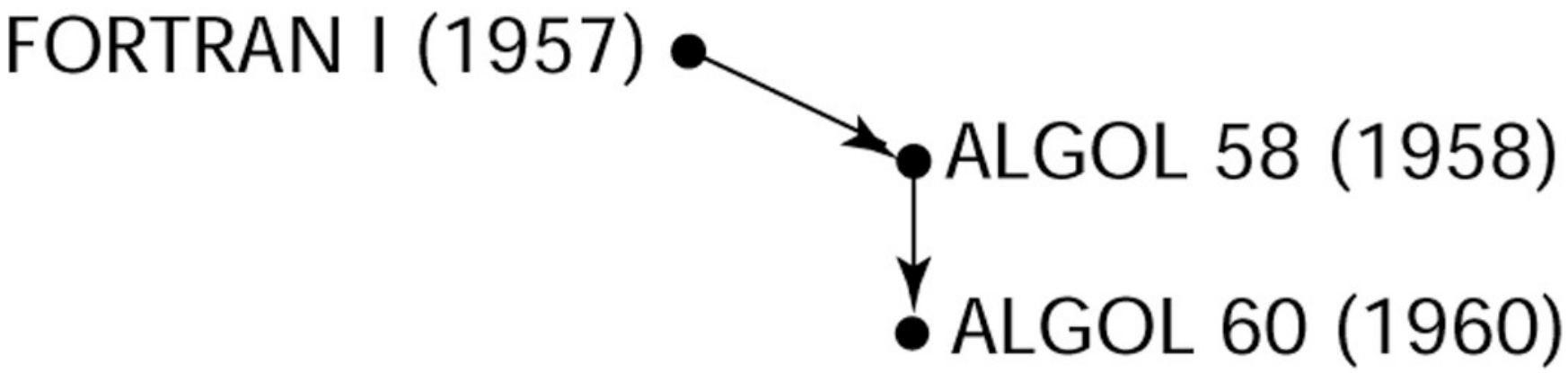
- Fortran IV – 1960 - 62
  - Declaração de tipos explícita
  - IF lógico (booleano)
  - Nomes de subprogramas como parâmetros a outros subprogramas
- Fortran 77 – 1978
  - Manuseio de cadeias de caracteres (strings)
  - Controle de laço IF com cláusula opcional ELSE
- Fortran 90 – 1990
  - Biblioteca de funções
  - Matrizes dinâmicas
  - Ponteiros
  - Recursão
  - Comando CASE
  - Verificação de tipos de parâmetros

# LISP - 1959

- LISt Processing language
  - Projetada no MIT por Jonh McCarthy
  - Inteligência Artificial
    - Processa dados em listas em vez de matrizes
    - Computação simbólica em vez de numérica
  - Possui somente dois tipos de dados: átomos e listas
  - Sintaxe baseada em Cálculo Lambda
  - Pioneira no paradigma de programação funcional
    - Não usa variáveis nem atribuições
    - Controle de fluxo de execução via recursão e expressões condicionais
  - Ainda é uma das principais linguagens em IA
  - Linguagens derivadas:
    - Common Lisp
    - Scheme
    - Miranda, ML e Haskell

# ALGOL 58 – 1958

- Fortran foi desenhada para a arquitetura IBM 70x
- Muitas linguagens começaram a surgir, todas para plataformas específicas
- Nenhuma delas era portável (machine dependent)
- Nenhuma linguagem universal para descrever algoritmos



# Algol 58

- Duas grandes associações, a ACM e a GAMM reuniram-se por 4 dias para especificar a linguagem
  - *Objetivos da linguagem:*
    1. Parecida com notação matemática
    2. Boa para descrever algoritmos
    3. Deve ser traduzível para código de máquina

- *Recursos da linguagem*:
  - Conceito de tipos foi formalizado
  - Nomes podem ter qualquer tamanho
  - Matrizes podem ter qualquer número de dimensões
  - Parâmetros foram separados por modo (in & out)
  - índices de matrizes forma colocados em colchetes
  - Comandos compostos (`begin ... end`)
  - ponto e vírgula como um separador de comandos
  - Operador de atribuição passou ser `:=`
  - `if` passou a ter uma cláusula `else-if`

- *Comentários:*
  - Não foi implementado da forma como foi especificada, mas sim variações dela: (MAD, JOVIAL)
  - Embora a IBM estivesse inicialmente entusiasmada, todo o suporte foi descontinuado em meados de 1959

# ALGOL 60 -1960

- Modificações no ALGOL 58 discutidas em uma reunião de 6 dias em Paris
  - *Novas funcionalidades*:
    - Estrutura de blocos (escopo local)
    - dois métodos de passagem de parâmetros
    - recursão em subprogramas
    - Matrizes dinâmicas
  - Ainda não havia manuseio de I/O e strings

  - *Sucessos*:
    - passou a ser o modo padrão de publicar algoritmos por mais de 20 anos
    - Todas as subseqüentes linguagens imperativas foram baseadas nela
    - Primeira linguagem independente de máquina
    - primeira linguagem cuja sintaxe foi formalmente definida (BNF)

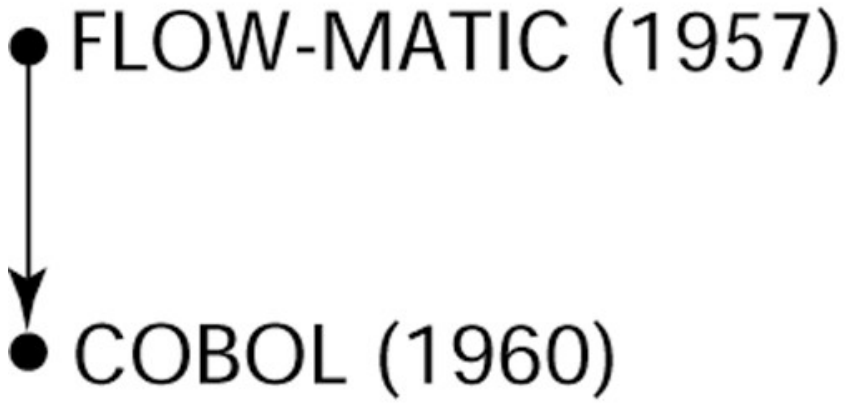
  - *Falhas*:
    - Nunca chegou a ser largamente utilizada, especialmente nos U.S.
      *Razões:*
      1. Nenhum suporte a i/o nem conjunto de caracteres faze com que os programas não sejam portáveis
      3. Muito flexível– difícil de implementar
      4. Entrincheiramento do FORTRAN
      5. descrição formal da sintaxe
      6. Falta de suporte da IBM

# COBOL – 1960

- ***Ambiente de desenvolvimento:***
  - **UNIVAC começou a usar FLOW-MATIC**
  - **USAF começou a usar AIMACO**
  - **IBM estava desenvolvendo COMTRAN**

- ***Baseada na linguagem FLOW-MATIC***
  - **características da FLOW-MATIC:**
    - **Nomes com até 12 caracteres, com hífens**
    - **nomes em inglês para operadores aritméticos**
    - **Dados e código eram completamente separados**
    - **Verbos eram a primeira palavra em cada comando**

- ***Primeira reunião para desenvolvimento da linguagem - maio 1959***
  - **Objetivos de projeto:**
    1. **Deveria parecer bastante com a língua inglesa**
    2. **Deveria ser fácil de usar, mesmo que isso significasse que a linguagem seria menos poderosa**
    3. **Deveria cobrir toda a base de usuários de computadores**
    4. **Não deveria ser polarizado pelos correntes problemas dos compiladores**
  - **Comitê de projeto foi formado por fabricantes de computadores e pelo Departamento de Defesa - DoD**
  - **Problemas de projeto: expressões aritméticas? subscritos? Brigas entre fabricantes**



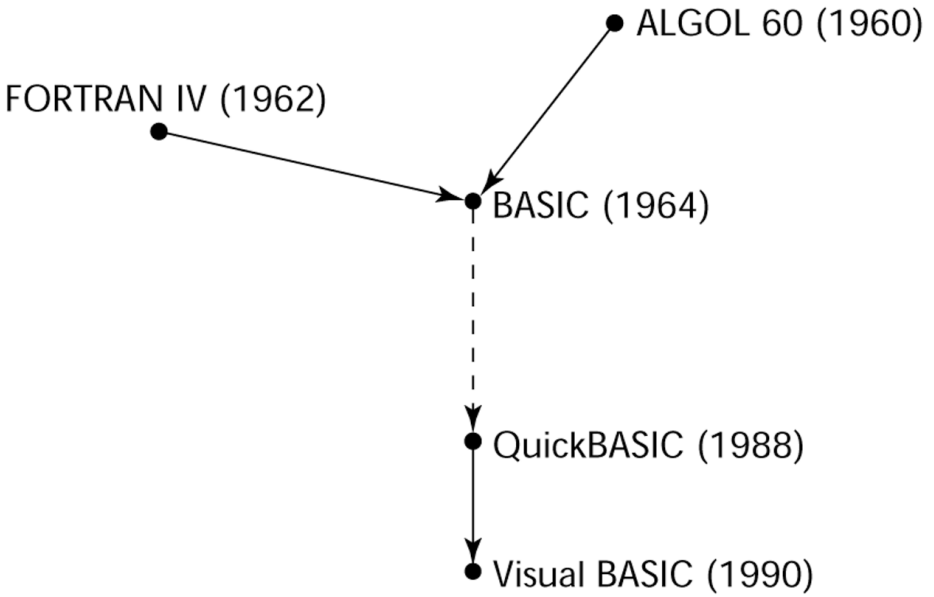
# COBOL – 1960

- ***Contribuições:***
  - **- Primeira construção em linguagem de alto nível para Macros**
  - **- Estruturas de dados hierárquicas (records)**
  - **- Comandos de seleção aninhados**
  - **- Nomes longos e conotativos (até 30 caracteres), com hífens**
  - **- Divisão de dados**

- ***- Comentários:***
  - **- Primeira linguagem especificada pelo DoD; Não teria sobrevivido se não fosse por isso**
  - **- É ainda a linguagem de aplicações comercias mas largamente utilizada**

# BASIC – 1964

- Projetada por Kemeny & Kurtz em Dartmouth
  - Objetivos de projeto:
    - Fácil de aprender e usar por estudantes de áreas não científicas
    - Deveria ser agradável e amigável
    - Retorno rápido para o trabalho de casa
    - Deveria permitir o acesso livre e privado
    - O tempo do usuário é mais importante do que o tempo da máquina
  - Dialetos atuais mais populares:
    QuickBASIC
    Visual BASIC



# PL/I – 1965

- Projetada pela IBM e pela SHARE (associação de usuários)
  - Situação da Computação em *1964 (ponto de vista da IBM)*
    1. Computação científica
       - computadores IBM 1620 e 7090
       - FORTRAN
       - grupo de usuários SHARE

    2. Computação comercial
       - computadores IBM 1401, 7080
       - COBOL
       - grupo de usuários GUIDE

- Em 1963, no entanto,
  - Usuários cientificos começaram a necessitar
    recursos de I/O mais elaborados, como no COBOL;
    usuários comerciais começaram a necessitar ponto flutuante e matrizes

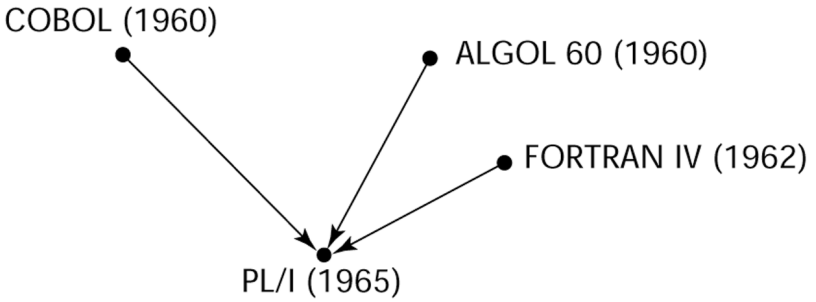
  - Parecia que muitas instalações passariam a necessitar de
    dois tipos de computadores, linguagens,
    e pessoal de suporte duplicado

- *A solução óbvia:*
  1. Construir um novo computador que atenda aos dois perfis
  2. Projetar uma nova linguagem para os dois tipos de aplicações

# PL/I

- ***Contribuições do PL/I:***
  1. **Concorrência**
  2. **Manuseio de exceção**
  3. **Recursão (opcional)**
  4. **Ponteiro como tipo de dado**
  5. **Seções transversais de matrizes**

- ***Comentários:***
  - **Construções mal projetadas**
  - **Muito grande e complexa**
  - **Foi (e ainda é) realmente usada em aplicações cientificas e comerciais**



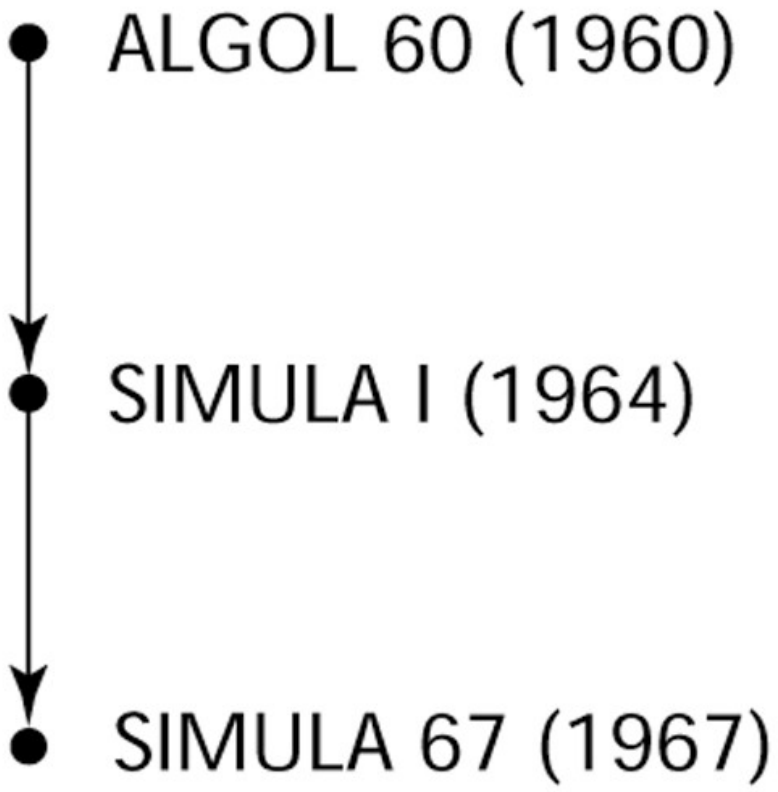
# Primeiras linguagens dinâmicas: APL e SNOBOL

- **- Caracterizadas por tipificação dinâmica e alocação de armazenagem dinâmica**

- **- APL (A Programming Language) 1962**
- **- Projetada como uma linguagem de descrição de hardware**
- **(na IBM por Ken Iverson)**
- **- Altamente expressiva (muitos operadores, para**
- **manipular escalares e matrizes de varias dimensões)**
- **- Programas são muito difíceis de ler**
- **- SNOBOL(1964)**
- **- Projetada como uma linguagem de manipulação de strings**
- **(na Bell por Farber, Griswold, e Polensky)**
- **- Operadores poderosos para reconhecimento de padrões de strings**

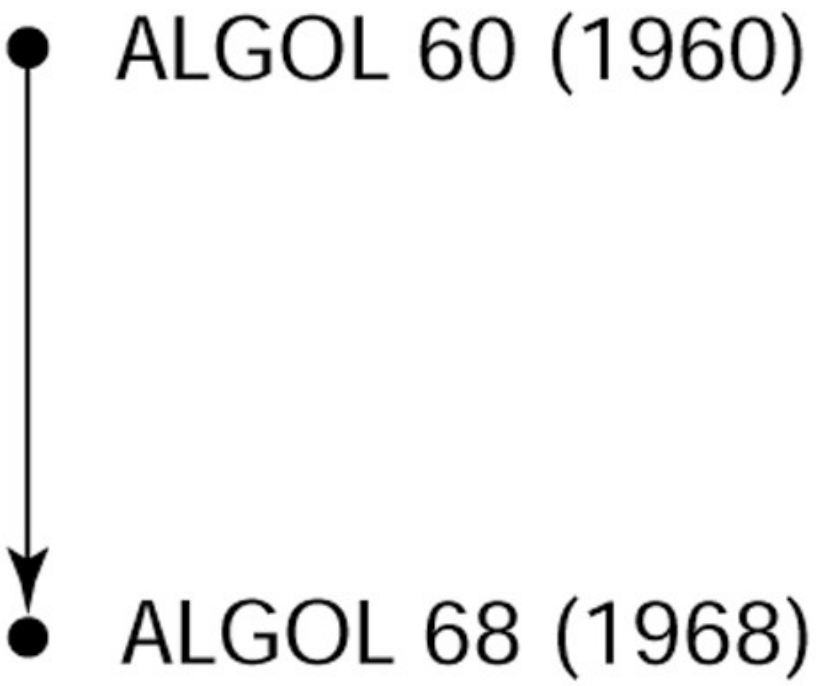
# Simula 67 – 1967

- Projetada inicialmente com propósito de simulação de sistemas na Noruega por Nygaard and Dahl
- Baseada nas linguagens ALGOL 60 e SIMULA I
- *Contribuições:*
  - Co-rotinas – uma espécie de sub-programa
  - Implementada em uma estrutura chamada classe
  - Classes são a base para o conceito de abstração de dados
  - Classes são estruturas que incluem dados locais e funcionalidades



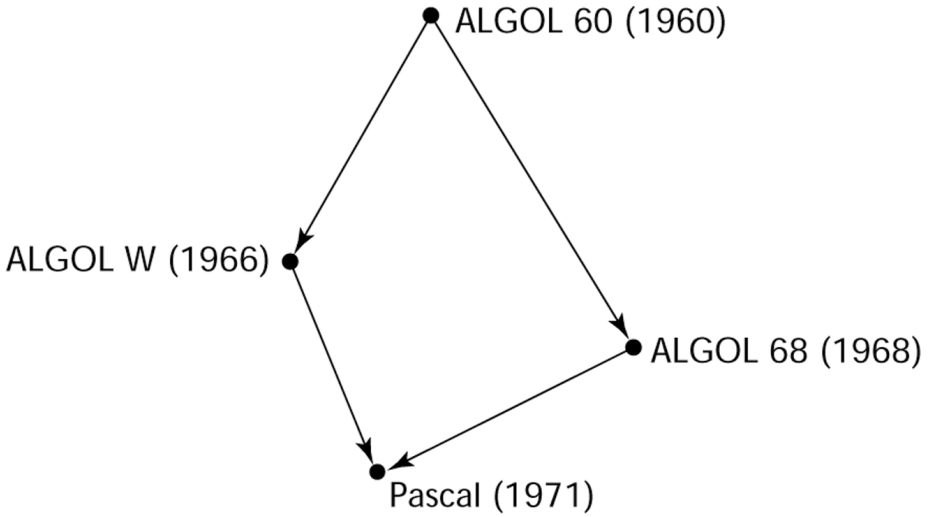
# Algol 68

- **Originou-se a partir da continuação do desenvolvimento da ALGOL 60,**
- **Mas não é uma mera extensão desta linguagem**
- **O projeto é baseado no conceito de ortogonalidade**
- ***Contribuições:***
  - **Estruturas de dados definidas pelo usuário**
  - **Tipos de referência**
  - **Matrizes dinâmicas**
- ***Comentários:***
  - **Foi menos usada que a ALGOL 60**
  - **Teve forte influência nas linguagens subseqüentes, especialmente Pascal, C, e Ada**



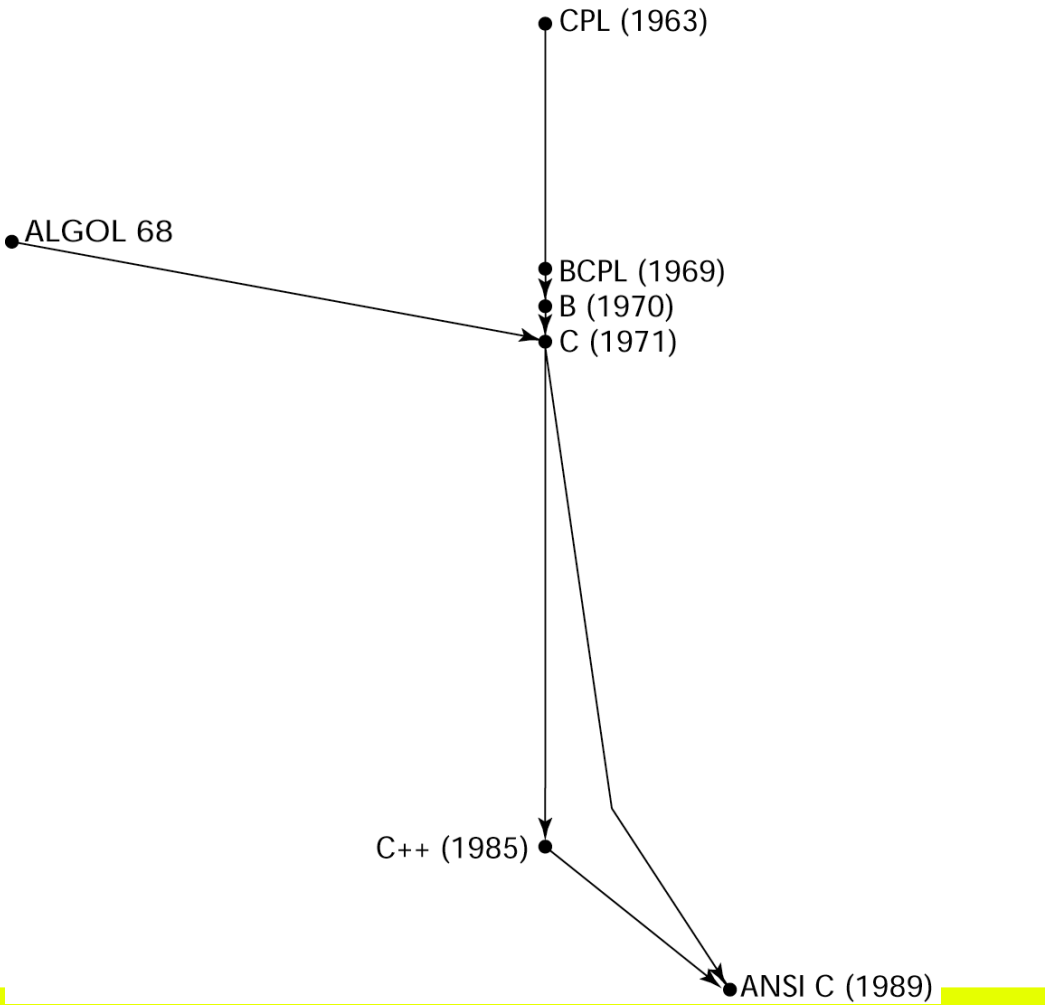
# Pascal

- Projetada por Niklaus Wirth, que deixou o comite do ALGOL 68 por discordar do trabalho
- Projetada para ensino de programação estruturada
- Pequena, simples, nada realmente novo
  - Ainda hoje a linguagem mais utilizada no ensino de programação nas universidades



# C

- Projetada para programação de sistemas nos laboratórios da Bell por Dennis Richie
- Evoluiu primariamente a partir da linguagem B mas também da ALGOL 68
- Possui um poderoso conjunto de operadores mas uma pobre verificação de tipos
- Teve sua utilização alavancada pelo sistema operacional UNIX



# Outras descendentes do Algol

- Modula-2 (meados dos anos 70 por Niklaus Wirth)
  - Baseado no Pascal com o acréscimo módulos and alguns recursos de programação em baixo nível para programação de sistemas
- Modula-3 (final dos anos 80 pela Digital & Olivetti)
  - Baseada no Modula-2 com adição de classes, manuseio de exceções, coleta de lixo e concorrência
- Oberon final dos anos 80 por Wirth
  - Adiciona suporte para OOP ao Modula-2
  - Muitos recursos do Modula-2 foram retirados
- Delphi (Borland)
  - Pascal adicionado a suporte a OOP
  - Mais elegante e seguro que C++

# Prolog – 1972

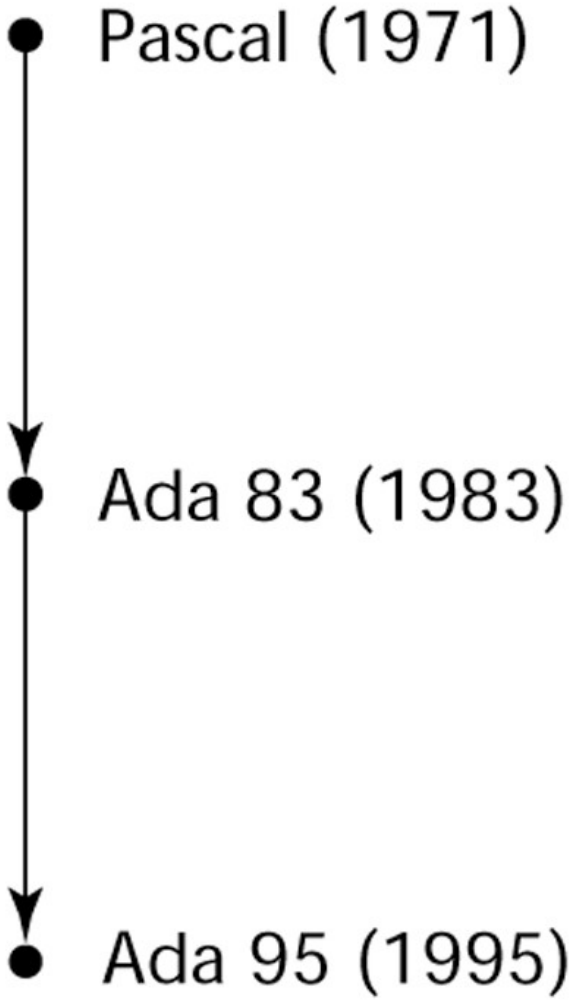
- Desenvolvido na University of Aix-Marseille por by Comerauer and Roussel, com ajuda de Kowalski daUniversity of Edinburgh
- Baseada e, lógica formal
- Não procedural
- Pode ser resumida como sendo umabase de dados inteligente que usa um processo de inferência para inferir a veracidade de uma consulta

# Ada - 1983

- Enorme esforço de projeto envolvendo centenas de pessoas, muito dinheiro e aproximadamente oito anos
- *Contribuições:*
  - Pacotes – suporte a abstração de dados
  - Manuseio de Exceções elaborado
  - Unidades genéricas de programas
  - Concorrência através de um modelo de tarefas

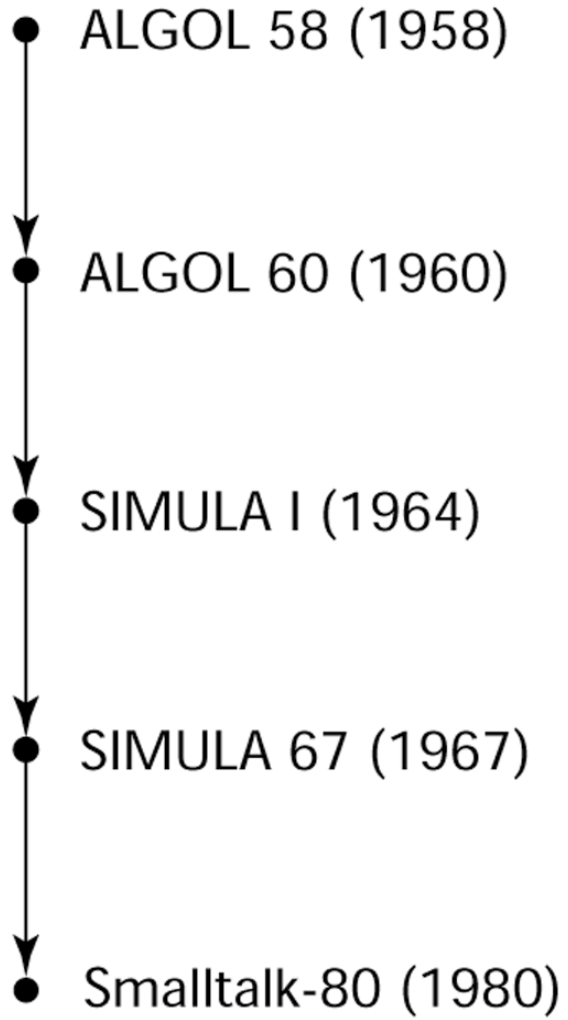
# Ada - 1983

- Projeto competitivo
- Inclui tudo que havia então  em engenharia de software e projeto de linguagens
- Os primeiros compiladores eram muito difíceis de implementar. O primeiro compilador realmente utilizável apareceu cindo anos depois
- Ada 95 (começou em 1988)
  - Suporte a OOP através da derivação de tipos
  - Melhores mecanismos de controle para compartilhamento de dados (novas facilidades pra concorrência)
  - Bibliotecas mais flexíveis



# Smaltalk – 1972 - 1980

- Desenvolvido pela Xerox PARC, initialmente por Alan Kay, depois por Adele Goldberg
- Primeira implementação completa de uma linguagem completamente orientada a objetos (abstração de dados, herança e ligação dinâmica de tipos)
- Pioneira em interface gráfica de usuário que todo mundo usa hoje



# C++ - 1985

- Desenvolvido pela Bell Labs por Bjarne Stroustrup
- Evoluiu a partir das linguagens C e SIMULA 67
- Recursos para programação orientada a objetos tirada parcialmente da SIMULA 67 foram adicionados ao C
- Possui também manuseio de exceções
- Uma linguagem grande e complexa, em parte porque suporta programação procedural e orientada a objetos
- Rapidamente cresceu em popularidade juntamente com OOP
- Padrão ANSI aprovado em novembro de 1997

# Eiffel

- Eiffel – é uma linguagem relacionada a família de linguagens que suportam OOP
- Projetada por Bertrand Meyer - 1992)
- Não foi diretamente derivada de nenhuma outra linguagem
- Menor e mais simples que C++, mas ainda assim possui a maior parte de sua capacidade

# Java - 1995

- Desenvolvido pela Sun no começo dos anos 90
- Baseada em C++
- Significativamente simplificada
- Suporta somente*y* OOP
- Possui  referencias, mas não ponteiros
- Inclui suporte a applets and uma forma de concorrência