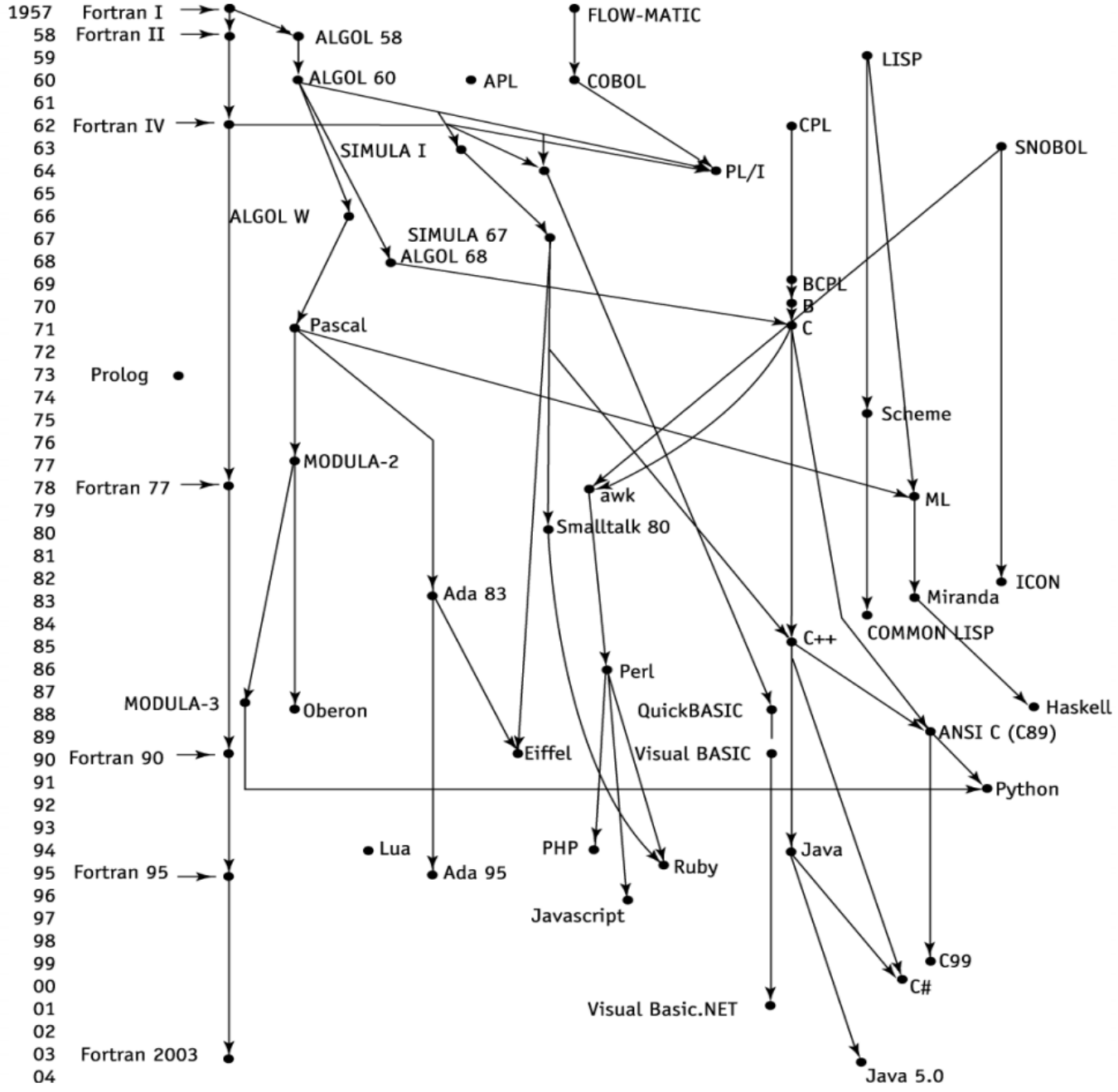
PARADIGMAS E LINGUAGENS DE PROGRAMAÇÃO

ENGENHARIA DA COMPUTAÇÃO – UFC/SOBRAL

Prof. Danilo Alves

danilo.alves@alu.ufc.br



- Genealogia das linguagens de programação

Feb 2020	Feb 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.358%	+1.48%
2	2		C	16.766%	+4.34%
3	3		Python	9.345%	+1.77%
4	4		C++	6.164%	-1.28%
5	7	^	C#	5.927%	+3.08%
6	5	v	Visual Basic .NET	5.862%	-1.23%
7	6	v	JavaScript	2.060%	-0.79%
8	8		PHP	2.018%	-0.25%
9	9		SQL	1.526%	-0.37%
10	20	^^	Swift	1.460%	+0.54%
11	18	^^	Go	1.131%	+0.17%
12	11	v	Assembly language	1.111%	-0.27%
13	15	^	R	1.005%	-0.04%
14	23	^^	D	0.917%	+0.28%
15	16	^	Ruby	0.844%	-0.19%
16	12	vv	MATLAB	0.794%	-0.40%
17	21	^^	PL/SQL	0.764%	-0.05%

PRIMEIRAS LINGUAGENS

UNIVERSIDADE FEDERAL DO CEARÁ

- Os primeiros computadores
  - lentos, caros, limitados, pouco confiáveis
  - ausência de software básico
- Programação trabalhosa
  - linguagem de máquina
  - codificação à mão
  - ausência de reutilização

PROGRAMAÇÃO DE HARDWARE (PSEUDOCÓDIGO)

UNIVERSIDADE FEDERAL DO CEARÁ

- O que estava errado ao usar código de máquina?
  - Baixa legibilidade
    - Programas eram escritos em linguagem de máquina
    - Não existia sequer linguagem de montagem
  - Modificações de programas tediosas e passíveis de erros
    - Problemas com referências internas a trechos do programa durante modificação ou exclusão de instruções (na exclusão, inserções NOP poderiam ser inseridas)
    - Esses problemas serviram de motivação para invenção de montadores e linguagem de montagem e mais recente, ligadores (linker)
  - Deficiências de máquina – sem indexação (vetores) ou ponto-flutuante

SHORT CODE

UNIVERSIDADE FEDERAL DO CEARÁ

- Short Code foi desenvolvida por John Mauchly em 1949 para o computador BINAC, era uma das principais maneiras de programar essas máquinas

– A descrição da linguagem original nunca foi publicada, seu conhecimento foi possível a partir de um manual de programação do UNIVAC I

– UNIVAC I possuía palavras de 72 bits – 12 bytes de 6 bits – Expressões foram codificadas

– Exemplo de operações:

```
01 -      06 abs value    1n (n+2)nd power      | X0 = SQRT(ABS(Y0))
02 )      07 +            2n (n+2)nd root       | 00 X0 03 20 06 Y0
03 =      08 pause        4n if <= n
04 /      09 (            58 print and tab
```

SPEEDCODING

UNIVERSIDADE FEDERAL DO CEARÁ

- Sistemas de interpretação para estender linguagens de máquina e incluir novas operações
- Speedcoding foi desenvolvido por John Backus em 1954 para o IBM 701
  - Pseudoinstruções para operações aritméticas e funções matemáticas em dados de ponto flutuante
  - Desvios condicionais e incondicionais
  - Memória usável restante após carregar o interpretador de apenas 700 palavras
  - Facilidade para incrementar os registradores de endereço automaticamente
- O interpretador convertia o 701 para uma calculadora virtual de ponto flutuante

RELATED SYSTEMS

- O sistema de "compilação" da UNIVAC
  - Desenvolvido por uma equipe liderada por Grace Hopper
  - Pseudocódigo expandido em código de máquina
- David J. Wheeler (Universidade de Cambridge)
  - Desenvolveu um método de usar blocos de endereços realocáveis para resolver parcialmente o problema do endereçamento absoluto

EVOLUÇÃO DE LINGUAGENS

UNIVERSIDADE FEDERAL DO CEARÁ

- Linguagem de máquina
  - falta de legibilidade
  - endereçamento absoluto
- Linguagem de montagem
  - montadores
  - uso de mnemônicos
  - endereçamento relativo

EVOLUÇÃO DE LINGUAGENS

UNIVERSIDADE FEDERAL DO CEARÁ

- Linguagens de alto nível
  - papel fundamental na evolução da computação
- Objetivos
  - ocultar hardware subjacente
  - aproximar-se dos domínios de aplicação
  - elevar o nível de abstração

EVOLUÇÃO DE LINGUAGENS - FORTRAN

UNIVERSIDADE FEDERAL DO CEARÁ

- Fortran 0: 1954 – não implementado
- Fortran I: 1957

  – Desenvolvido para o IBM 704, que tinha registros de indexação e hardware de ponto-futuante

  – Levou à ideia de linguagens de programação compiladas, porque não havia o esconderijo para o custo da interpretação

  – Ambiente de desenvolvimento utilizado na época

  - Computadores com memórias pequenas e não confiáveis
  - Aplicações eram científicas
  - Não havia maneiras eficientes de programar computadores
  - Velocidade do código objeto era o objetivo principal

EVOLUÇÃO DE LINGUAGENS – FORTRAN I

UNIVERSIDADE FEDERAL DO CEARÁ

- Primeira versão implementada do FORTRAN
  - Formatação de entrada e saída
  - Nomes de variáveis podem ter até seis caracteres (eram dois no FORTRAN 0)
  - Sentença de seleção (IF)
  - Sentença de repetição (DO)
  - Sub-rotinas definidas pelos usuários, sem compilação separada
  - Compilador lançado em abril de 1957, depois de 18 anos de trabalho
  - Programas com mais de 400 linhas raramente são compilados corretamente, especialmente devido à pouca confiabilidade do 704
  - O código era muito rápido
  - Rapidamente se tornou amplamente usado

EVOLUÇÃO DE LINGUAGENS – FORTRAN II

UNIVERSIDADE FEDERAL DO CEARÁ

- Distribuído em 1958
  - Compilação independente de sub-rotinas
    - Sem a compilação independente, sempre que houvesse mudanças no programa, todo o código deveria ser recompilado
  - Corrigiu falhas da versão anterior do compilador

EVOLUÇÃO DE LINGUAGENS – FORTRAN IV

UNIVERSIDADE FEDERAL DO CEARÁ

- Evoluiu entre 1960-62
  - Declarações de tipo explícitas
  - Construção **if** lógica
  - Nomes de subprogramas podem ser parâmetros
  - Padrão ANSI em 1966

EVOLUÇÃO DE LINGUAGENS – FORTRAN 77

UNIVERSIDADE FEDERAL DO CEARÁ

- Tornou-se o novo padrão em 1978
  - Manipulação de cadeias de caracteres
  - Sentenças de controle de laços lógicos
  - Um **If** com uma cláusula opcional **Else**

EVOLUÇÃO DE LINGUAGENS – FORTRAN 90

UNIVERSIDADE FEDERAL DO CEARÁ

- Drasticamente diferente do Fortran 77
  - Módulos
  - Vetores dinâmicos
  - Ponteiros
  - Registros
  - Sentença CASE
  - Recursão
- Aviso de remoção de recursos obsoletos de versões anteriores

EVOLUÇÃO DE LINGUAGENS – FORTRAN

UNIVERSIDADE FEDERAL DO CEARÁ

- Compiladores altamente otimizados (todas as versões antes de 90)
  - Tipos e armazenamento para todas as variáveis são fixados antes da execução
- Mudou drasticamente para sempre a forma como os computadores são usados
- Caracterizada como a língua franca do mundo da computação

EVOLUÇÃO DE LINGUAGENS – FORTRAN

UNIVERSIDADE FEDERAL DO CEARÁ

	Formato	Uso
	Iw[.m]	Valores Inteiros
	Fw.d	Valores Reais
	Ew.d[Ee]	Valores Reais com expoente
	Gw.d[Ee]	Mesmo que Iw[.m], Ew.d[Ee], Lw e A[w]
	Dw.d	Valores Reais de Dupla Precisão
	Lw	Valores Lógicos
	A[w]	Seqüência de Caracteres
N77	Zw_hexedit	Valores Hexadecimais
F90	B*w*[.*m*]	Valores Binários
F90	O*w*[.*m*]	Valores Octadecimais
F90	EN*w*.*d*[E*e*]	Valores Reais em Notação de Engenharia
F90	ES*w*.*d*[E*e*]	Valores Reais em Notação Cientifica

FORTRAN	F90	Matemática Tradicional	Significado
.LT.	<	<	MENOR QUE
.LE.	<=	£	MENOR OU IGUAL QUE
.EQ.	==	=	IGUAL A
.NE.	/=	1	DIFERENTE DE
.GT.	>	>	MAIOR QUE
.GE.	>=	3	MAIOR OU IGUAL QUE

EVOLUÇÃO DE LINGUAGENS – FORTRAN

UNIVERSIDADE FEDERAL DO CEARÁ

```
C     FIND THE MEAN OF N NUMBERS AND THE NUMBER OF
C     VALUES GREATER THAN IT

      DIMENSION A(99)
      REAL MEAN
      READ(1,5)N
 5    FORMAT(I2)
      READ(1,10)(A(I),I=1,N)
 10   FORMAT(6F10.5)
      SUM=0.0
      DO 15 I=1,N
 15      SUM=SUM+A(I)
      MEAN=SUM/FLOAT(N)
      NUMBER=0
      DO 20 I=1,N
         IF (A(I) .LE. MEAN) GOTO 20
         NUMBER=NUMBER+1
 20   CONTINUE
      WRITE (2,25) MEAN,NUMBER
 25   FORMAT(11H MEAN = ,F10.5,5X,21H NUMBER SUP = ,I5)
      STOP
      END
```

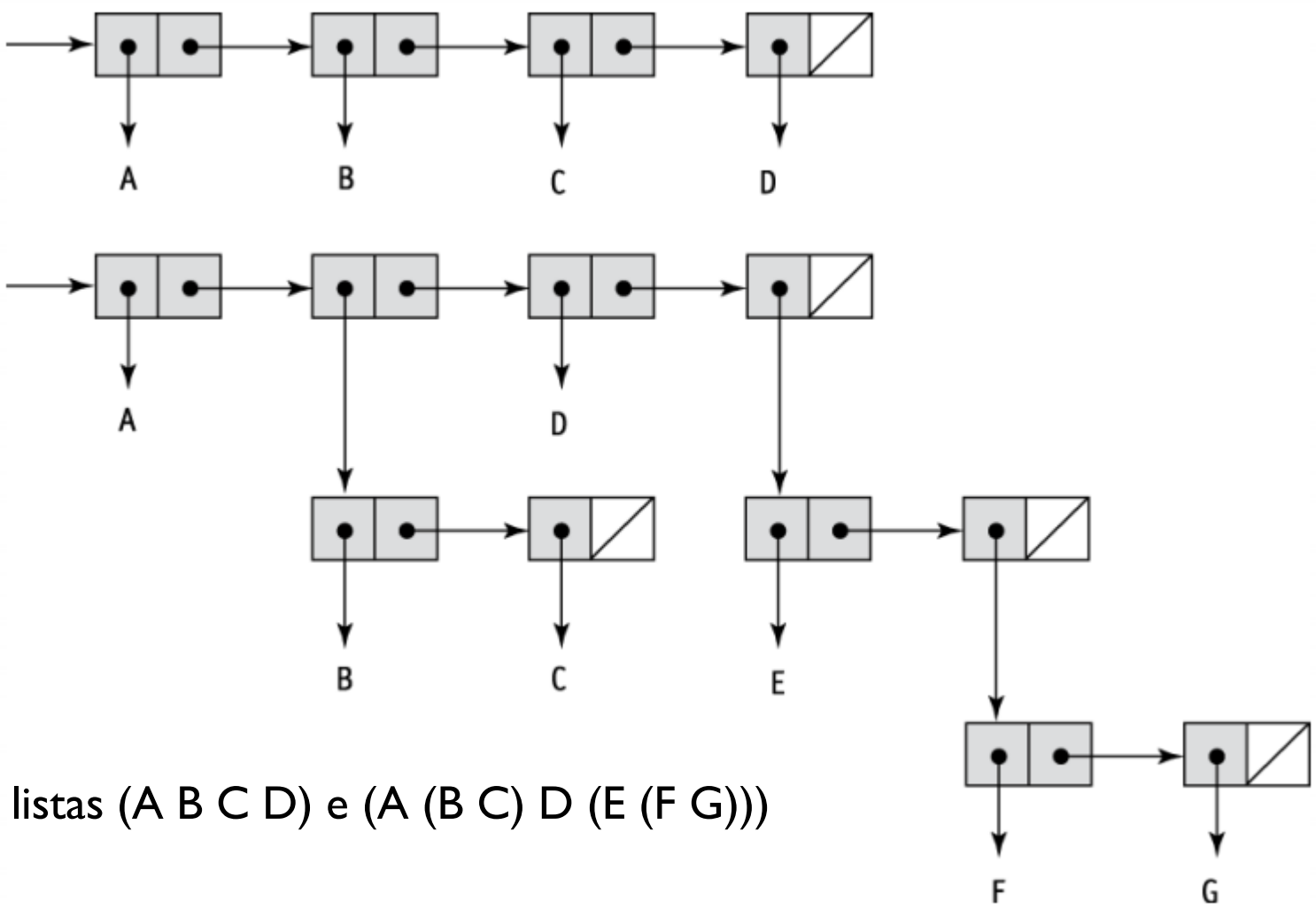
PROGRAMAÇÃO FUNCIONAL: LISP

UNIVERSIDADE FEDERAL DO CEARÁ

- List Processing Language
  - Projetada no MIT por McCarthy
- Pesquisa de inteligência artificial (IA) precisava de uma linguagem para
  - Processar dados em listas (em vez de vetores)
  - Computação simbólica (em vez de numérica)
- Apenas dois tipos de dados: átomos e listas
- Sintaxe é baseada em *lambda calculus*

```
((lambda (arg) (+ arg 1)) 5)
```

REPRESENTAÇÃO INTERNA DE DUAS LISTAS EM LISP



- Representando as listas (A B C D) e (A (B C) D (E (F G)))

AVALIAÇÃO DE LISP

UNIVERSIDADE FEDERAL DO CEARÁ

- Pioneira na programação funcional
  - Sem necessidade de variáveis ou atribuição
  - Controle por recursão e expressões condicionais

- Ainda a linguagem dominante em IA
- COMMON LISP e Scheme são dialetos contemporâneos de LISP
- ML (MetaLanguage), Miranda e Haskell são linguagens relacionadas

SCHEME

UNIVERSIDADE FEDERAL DO CEARÁ

- Desenvolvida no MIT no meio dos anos 1970
- Pequena
- Uso de escopo estático
- Funções como entidades de primeira classe
- Sintaxe simples (e tamanho pequeno) o que a torna ideal para aplicações educacionais

COMMON LISP

UNIVERSIDADE FEDERAL DO CEARÁ

- Um esforço para combinar características de diversos dialetos de LISP em uma só linguagem
- Grande, complexa

```
(defun ...)
; testes
(print
    (filter
        (lambda (x) (if (< x 10) x nil)
        '(3 12 6 15 9))))
```

O PRIMEIRO PASSO EM DIREÇÃO À SOFISTICAÇÃO: ALGOL 60

UNIVERSIDADE FEDERAL DO CEARÁ

- Ambiente de desenvolvimento
  - FORTRAN chegou (apenas) para IBM 700
  - Várias outras linguagens foram desenvolvidas, todas para máquinas específicas
  - Nenhuma linguagem portátil, todas eram dependentes das máquinas
  - Nenhuma linguagem universal para comunicação de algoritmos
- ALGOL 60 foi o resultado dos esforços para criar uma linguagem universal

PROCESSO DO PROJETO INICIAL

UNIVERSIDADE FEDERAL DO CEARÁ

- ACM e GAMM se reuniram por quatro dias (De 27 de maio a 1° de junho de 1958)
- Definir uma linguagem cujos objetivos eram
  - Ser o mais próxima possível da notação padrão matemática
  - Boa para a descrição de algoritmos
  - Ser traduzível em código de máquina

ALGOL 58

UNIVERSIDADE FEDERAL DO CEARÁ

- Formalizou conceito de tipo de dados
- Identificadores podiam ter qualquer tamanho
- O vetor podia ter qualquer dimensão
- O limite inferior dos vetores podia ser especificado pelo programador
- Sentenças de seleção aninhadas eram permitidas
- Declarações compostas (**begin ... end**)
- Vírgula como separador de declarações
- Operador de atribuição era **:=**
- **if** tinha uma cláusula **else-if**

VISÃO GERAL DO ALGOL 60

UNIVERSIDADE FEDERAL DO CEARÁ

- Modificação do ALGOL 58 em seis dias de encontros em Paris
- Novos recursos
  - Estrutura de bloco (escopo local)
  - Duas formas diferentes de passagem de parâmetros a subprogramas
  - Procedimentos recursivos
  - Vetores dinâmicos na pilha
  - Ainda sem sentenças de entrada e saída e sem manipulação de cadeias

AVALIAÇÃO DO ALGOL 60

UNIVERSIDADE FEDERAL DO CEARÁ

- Sucessos
  - Única maneira formal aceitável de comunicar algoritmos por mais de 20 anos
  - Todas as linguagens de programação imperativas desde 60 são baseadas nela
  - Primeira linguagem independente de máquina
  - Primeira linguagem cuja sintaxe foi formalmente descrita (BNF)

AVALIAÇÃO DO ALGOL 60

- Fracassos
  - Nunca atingiu uso disseminado, especialmente nos Estados Unidos
  - Motivos
    - Entrada e saída tiveram uma portabilidade ruim
    - Muito flexível – difícil de implementar
    - Forte estabelecimento do Fortran
    - Descrição formal de sintaxe (estranha e complicada para a época)
    - Falta de apoio da IBM

INFORMATIZANDO OS REGISTROS COMERCIAIS: COBOL

UNIVERSIDADE FEDERAL DO CEARÁ

- Ambiente de desenvolvimento
  - UNIVAC estava começando a usar FLOW-MATIC
  - USAF estava começando a usar AIMACO – variação do FLOW-MATIC
  - IBM estava desenvolvendo COMTRAN – nunca foi implementada

PERSPECTIVA HISTÓRIA DO COBOL

UNIVERSIDADE FEDERAL DO CEARÁ

- Baseado em FLOW-MATIC
- Características de FLOW-MATIC
  - Nomes com mais de 12 caracteres, com hifens
  - Nomes em inglês para operações aritméticas
  - Dados e código eram completamente separados
  - A primeira palavra em cada sentença era um verbo

O PROCESSO DO PROJETO DO COBOL

UNIVERSIDADE FEDERAL DO CEARÁ

- Primeira reunião (no Pentágono) – Maio de 1959
- Objetivos
  - Utilizar o inglês o máximo possível
  - Ser fácil de usar, mesmo ao custo de ser menos poderosa
  - Ampliar a base de usuários de computador
  - Não ser orientada pelos problemas atuais do compilador

AVALIAÇÃO DO COBOL

UNIVERSIDADE FEDERAL DO CEARÁ

- Contribuições
  - Primeira construção para macros de uma linguagem de alto nível
  - Estruturas de dados hierárquicas (registros)
  - Sentenças de seleção aninhadas
  - Nomes longos (até 30 caracteres), com hifens
  - Divisão de dados – número de dígitos decimais, localização do ponto decimal
  - Registros de arquivos – ideal para relatórios

COBOL: INFLUÊNCIA DO DOD

UNIVERSIDADE FEDERAL DO CEARÁ

- Primeira linguagem requerida pelo DoD – Uso obrigatório pelo DoD
  - Teria falhado sem o DoD
- Ainda assim, a linguagem mais utilizada em aplicações de negócios

O INÍCIO DO COMPARTILHAMENTO DE TEMPO: BASIC

UNIVERSIDADE FEDERAL DO CEARÁ

- Projetado por Kemeny e Kurtz em Dartmouth
- Objetivos:
  - Ser fácil de aprender, para estudantes que não são de ciências básicas
  - Ser prazerosa e amigável
  - Agilizar os deveres de casa
  - Permitir acesso livre e privado
  - Considerar o tempo do usuário mais importante do que o tempo do computador
- Ignorada por cientistas da computação, perdeu espaço com o surgimento de outras LP
- Dialeto popular à época: Visual BASIC (1990), que revitalizou a linguagem

O INÍCIO DO COMPARTILHAMENTO DE TEMPO: BASIC

- Primeira linguagem amplamente utilizada com o tempo de compartilhamento
  - Permitia o acesso individual através de terminais a vários usuários ao mesmo tempo

TUDO PARA TODOS: PL/I – *PROGRAMMING LANGUAGE ONE*

UNIVERSIDADE FEDERAL DO CEARÁ

- Desenvolvida por IBM e SHARE (Comitê de desenvolvimento de LP)
- Situação da computação em 1964 (do ponto de vista da IBM)
  - Aplicação científica
    - Computadores IBM 1620 e 7090
    - FORTRAN
    - Grupo de usuário SHARE
  - Aplicação de negócios
    - Computadores IBM 1401 e 7080
    - COBOL
    - Grupo de usuário GUIDE

PERSPECTIVA HISTÓRICA

UNIVERSIDADE FEDERAL DO CEARÁ

- Em 1963

  – Programadores científicos passaram a precisar de recursos mais elaborados de entrada e saída, como COBOL tinha; as aplicações de negócios precisavam de dados de ponto-futuante e vetores para sistemas de informação de gerenciamento

  – Começou a parecer que as instalações de computação logo precisariam de duas equipes técnicas e de computadores diferentes

- A solução óbvia

  – **Construir um novo computador para fazer os dois tipos de aplicações**

  – **Projetar uma nova linguagem para as aplicações**

O PROCESSO DE PROJETO

UNIVERSIDADE FEDERAL DO CEARÁ

- Desenvolvido em cinco meses pelo Comitê 3 x 3
  - Três membros da IBM, três membros do SHARE
- Projeto inicial
  - Uma extensão do Fortran IV
- Inicialmente chamado de NPL (*New Programming Language* - Nova Linguagem de Programação)
- Nome mudado para PL/I em 1965 para evitar confusão com o NPL de *National Physical Laboratory*, na Inglaterra

AVALIAÇÃO DE PL/I

UNIVERSIDADE FEDERAL DO CEARÁ

- Contribuições
  - Permitido aos programas criar subprogramas executados concorrentemente
  - Possível detectar e manipular exceções
  - Permitida a utilização de subprogramas recursivamente
  - Ponteiros foram incluídos como um tipo de dados
  - Porções de uma matriz podiam ser referenciadas
- Preocupações
  - Muitos dos novos recursos foram mal concebidos
  - Muito grande e muito complexo

O INÍCIO DA ABSTRAÇÃO DE DADOS: SIMULA 67

UNIVERSIDADE FEDERAL DO CEARÁ

- Projetada inicialmente para simulação, na Noruega, por Nygaard e Dahl
- Baseada no ALGOL 60 e no SIMULA I
- Contribuições
  - Corrotinas – espécie de subprograma, que permite diferentes entradas/saídas da subrotina
  - **Classes**, **objetos** e **herança**

O INÍCIO DA ABSTRAÇÃO DE DADOS: SIMULA 67

UNIVERSIDADE FEDERAL DO CEARÁ

- Projetada na IBM por Ken Iverson, em torno de 1960, como uma linguagem para descrever arquiteturas de computadores (e não como uma linguagem implementada)

  – Alta expressividade (grande número de operadores, grande número de operações unitárias em vetores)

  – Programas difíceis de ler
- Ainda em uso;
- Mudanças mínimas

PROJETO ORTOGONAL: ALGOL 68

UNIVERSIDADE FEDERAL DO CEARÁ

- A partir do desenvolvimento continuado do ALGOL 60
- Fonte de uma série de novas ideias (embora a própria linguagem nunca tenha alcançado grande uso)
- Projeto baseado no conceito de ortogonalidade
  - Alguns conceitos primitivos mais o uso irrestrito de mecanismos de combinação

AVALIAÇÃO DO ALGOL 68

UNIVERSIDADE FEDERAL DO CEARÁ

- Contribuições
  - Estruturas de dados definidas pelo usuário
  - Tipos de referência
  - Vetores dinâmicos
- Comentários
  - Menos uso do que o ALGOL 60
  - Teve forte influência nas linguagens subsequentes, especialmente Pascal, C e Ada

PASCAL - 1971

- Projetada por Niklaus Wirth (ex-membro do comitê do ALGOL 68) para ser usada como veículo educacional
- Pequena, simples, nada realmente novo
- O maior impacto foi no ensino de programação

  – Do meio dos anos 1970 até o fm dos 1990, foi a linguagem mais usada para o ensino de programação

C - 1972

UNIVERSIDADE FEDERAL DO CEARÁ

- Projetada para a programação de sistemas (no Bell Labs, por Dennis Richie)
- Evoluída a partir de BCLP, B e ALGOL 68
- Poderoso conjunto de operadores
- Primeira linguagem de alto padrão implementada no UNIX
- Muitas áreas de aplicação

O MAIOR ESFORÇO DE PROJETO DA HISTÓRIA: ADA

UNIVERSIDADE FEDERAL DO CEARÁ

- Enorme esforço de projeto, envolvendo centenas de pessoas, muito dinheiro e cerca de 8 anos
  - Strawman (abril de 1975)
  - Woodman (agosto de 1975)
  - Tinman (1976)
  - Ironman (1977)
  - Steelman (1978)
- Chamada de Ada em homenagem de

Augusta Ada Byron, a primeira programadora

ADA – CURIOSIDADE

- Seu nome de nascimento é Augusta Ada Byron, flha do poeta Lord Byron
- Quando se casou, em 1835, passou a ser Augusta Ada King – seu marido era o Barão Willian King

  – Três anos depois ele se tornou o Conde de Lovelace e Ada também ganhou o título, passando a ser a Condessa de Lovelace
- O seu amigo e mentor Babbage a apelidou de "Feiticeira dos Números"

AVALIAÇÃO DE ADA

- Contribuições
  - Pacotes – suporte para abstração de dados
  - Recursos elaborados para manipulação de exceções
  - Unidades de programas genéricas
  - Concorrência de unidades de programas especiais
- Comentários
  - Projeto competitivo
  - Agrupa a maioria dos conceitos de engenharia de software e projeto de linguagem do final dos anos 1970
  - Primeiros compiladores eram difíceis; apenas em 1985, quase quatro anos após o projeto da linguagem estar completo, compiladores usáveis começaram a aparecer

PROGRAMAÇÃO ORIENTADA A OBJETOS: SMALLTALK

UNIVERSIDADE FEDERAL DO CEARÁ

- Projetada na Xerox PARC (Palo Alto Research Center), inicialmente por Alan Kay, depois por Adele Goldberg
- **Primeira linguagem de programação a oferecer suporte completo à programação orientada a objetos**
- **Pioneira no design da interface gráfica do usuário**

IMPERATIVIDADE E PROGRAMAÇÃO ORIENTADA A OBJETOS: C++

UNIVERSIDADE FEDERAL DO CEARÁ

- Projetada no Bell Labs por Stroustrup em 1980
- Desenvolvida a partir de C e SIMULA 67
- Facilidades para programação orientada a objetos, emprestadas do SIMULA 67
- Fornece manipulação de exceções
- Linguagem grande e complexa, em parte porque suporta programação procedural e orientada a objetos
- Cresceu rapidamente em popularidade
- Padrão ANSI aprovado em novembro de 1997
- Versão da Microsoft (lançado com .NET em 2002): *Managed* C++
  - .NET não suporta herança múltipla, logo MC++ também não o faz

UMA LINGUAGEM ORIENTADA A OBJETOS BASEADA NO PARADIGMA IMPERATIVO: JAVA

UNIVERSIDADE FEDERAL DO CEARÁ

- Projetada na Sun no início dos anos 1990
  - C e C++ não eram satisfatórios para dispositivos eletrônicos embarcados
- Baseada em C++
  - Significantemente simplificada
  - Suporta apenas programação orientada a objetos
  - Tem referências, mas não tem ponteiros
  - Inclui forma simples de controle de concorrência

AVALIAÇÃO DE JAVA

UNIVERSIDADE FEDERAL DO CEARÁ

- Eliminou muitos recursos inseguros de C++, como ponteiros
- Suporta concorrência, com métodos simples de controle
- Bibliotecas de classes para interfaces gráficas com o usuário, acesso a bases de dados e redes
- Portabilidade: Máquina Virtual Java (JVM), compiladores Just-in-Time (JIT)
- Amplamente usado para programação Web
- Uso aumentou mais rapidamente do que qualquer linguagem anterior
- Versão mais recente Java12

LINGUAGENS DE SCRIPTING PARA WEB

UNIVERSIDADE FEDERAL DO CEARÁ

- Perl
  - Desenvolvida por Larry Wall — lançada primeiro em 1987
  - Variáveis são estaticamente tipadas e implicitamente declaradas
  - Três espaços de nomes distintos para variáveis, denotados pelo primeiro caractere de nomes de variáveis
  - Teve uso difundido para programação CGI na Web
  - Também usado como ferramenta de administração de sistema em UNIX
- JavaScript
  - Começou na Netscape, mas depois se tornou um projeto conjunto da Netscape com a Sun Microsystems
  - Permitia aos documentos HTML requisitarem a execução de programas no servidor; usado na criação de documentos HTML dinâmicos
  - Puramente interpretada
  - Relacionado ao Java somente por meio de sintaxe similar

LINGUAGENS DE SCRIPTING PARA WEB

UNIVERSIDADE FEDERAL DO CEARÁ

- PHP
  - PHP: Hypertext Preprocessor (Processador de Hipertexto), projetado por Rasmus Lerdorf
  - Linguagem de scripting do lado do servidor embutida em HTML, geralmente utilizados para processamento de formulários e acesso de dados pela Web
  - Puramente interpretada
  - Suporte à orientação a objetos a partir da versão 3
- Python
  - Linguagem de *scripting* orientada a objetos
  - Com verificação de tipos, mas tipada dinamicamente
  - Suporta listas e dicionários

LINGUAGENS DE SCRIPTING PARA WEB

UNIVERSIDADE FEDERAL DO CEARÁ

- Ruby
  - Projetada no Japão por Yukihiro Matsumoto (também conhecido como "Matz")
  - Começou como um substituto para Perl e Python
  - Linguagem de scripting orientada a objetos pura
  - Todos os dados são objetos
  - A maioria dos operadores são implementados como métodos, que podem ser redefinidos pelo código do usuário
  - Puramente interpretada