**O CONTEÚDO ESCRITO ABAIXO É APENAS UM RESUMO DOS MEUS ESTUDOS A RESPEITO DE CADA TÓPICO.**

**Linguagens de programação**

**Evolução até o advento do computador**

**Primórdios**:

1. 2000 a.C.

Ábaco.

1. 1614

John Napier - logaritmos;

Redução de multiplicação e divisão a adição e subtração.

1. 1623

Francis Bacon - aritmética de base 2.

1. 1624

Heirich Schikart - primeira máquina com rodas dentadas.

1. 1642

Blaise Pascal - máquina de somar: primeira calculadora mecânica.

1. 1670

Gottofried Leibniz - máquina de quatro operações.

1. 1728

J. Falcon - cartões perfurados.

1. 1805

Joseph Marie Jacquard - tear automático.

1. 1830

Fabricação de máquinas de calcular em série.

**Amadurecimento**:

1. 1834

Charles Babbage - máquina analítica.

1. 1835

Ada Lovelace - primeiro programa.

1. 1847

George Boole - lógica matemática.

1. 1890

Herman Hollerith - processamento de dados automatizados.

1. 1900

Valdemar Pouslen - registros magnéticos.

1. 1906

Lee de forest - válvula.

1. 1936

Alan Turing - fundamentação teórica de um computador.

1. 1938

Claude Shannon - operações lógicas com circuitos eletrônicos.

**Último século**:

1. 1941

Konrad Zuse - primeiro computador eletrônico programável.

1. 1943

John Von Neumann - programa armazenado em memória.

1. 1944

Universidade de Harvard - primeiro computador com características modernas.

1. 1946

Presper Eckter e John Marchly - primeiro computador com válvulas (ENIAC).

**Software**

A parte física do computador é denominada **hardware**. Aos programas, essenciais para sua conveniente utilização dos problemas, chamamos **software**. Ao pessoal, capaz de conduzir hardware e software, denominamos **peopleware.**

* **Software do fabricante:** Programas preparados pelo fabricante (hardware ou software) para simplificar a operação do computador.
* **Software do usuário:** Programas preparados pelo usuário para atingir os objetivos específicos de sua organização.

No software do fabricante destaca-se o sistema operacional. O **sistema operacional** provê:

* administração de arquivos;
* controle dos periféricos;
* execução de utilitários.

**Linguagens de programação**

**Linguagem de programação** é um conjunto de termos (vocabulário) e de regras (sintaxe) que permite a formulação de instruções a um computador.

No início foi extremamente difícil passar um programa a uma máquina porque não se concebera um meio de controlar uma barreira presente em qualquer computador: a sua limitação de só reconhecer “instruções” colocadas em sua memória sob forma de dígitos binários. Assim, ao programar em linguagem de máquina, no seu nível mais elementar, o programador utilizava apenas linguagem binária, isto é, usava apenas zeros (0) e uns (1).

**Linguagem Assembly**

É uma linguagem orientada para máquina, cujas instruções têm geralmente uma correspondência um-para-um com as instruções de máquina, e que pode permitir facilidades tais como o uso de microinstruções. Para ser executada pelo computador, necessita passar pela fase de montagem (*assembler*) através de um montador (Assembler, Assembler Program), que traduz suas instruções para instruções de máquinas executáveis pelo computador.

**Linguagem COBOL**

É uma linguagem de programação clássica, originada na Terceira Geração. O nome, sigla de Common Business Oriented Language (Linguagem Orientada aos Negócios), define seu objetivo principal: aplicação em sistemas comerciais, financeiros e administrativos para empresas e governos. Tem sido permanentemente atualizado: o **COBOL 2002** já inclui suporte para programação orientada a objetos e outras características da linguagem moderna.

Criado em 1959 durante a CODASYL (Conference on Data Systems Language), um dos três comitês propostos em uma reunião no Pentágono em maio daquele ano, foi lançado comercialmente em agosto de 1961.

**Linguagem C**

É uma linguagem de programação compilada, de propósito geral, estruturada, imperativa, procedural, de alto nível, padronizada pela ISO, criada por Dennis Ritchie, no AT&T Bell Labs, para desenvolver o sistema operacional Unix (originalmente escrito em Assembly).

**C** é uma das linguagens de programação mais populares; são poucas as arquiteturas para as quais não existem compiladores para ela. Tem influenciado muitas outras linguagens, mais notavelmente **C++,** que começou como uma extensão sua.

**Linguagem C++**

Conhecido popularmente como “cê mais mais”, é uma linguagem de programação multiparadigma e de uso geral. É considerada de médio nível, pois combina características de linguagens de alto e baixo níveis. Desde os anos 1990, é uma das linguagens comerciais mais populares, sendo bastante usada academicamente, por seu grande desempenho e base de utilizadores.

Desenvolvida em 1983, no Bell Labs, por Bjarne Stroustrup foi originalmente conhecida como *C with classes* (C com classes) e era um adicional à linguagem C. novas características lhe foram sendo adicionadas, como tratamento de exceções.

**Linguagem C#**

Conhecida como **C Sharp**, é uma linguagem de programação orientada a objetos, fortemente tipada, desenvolvida pela Microsoft como parte da plataforma dotNet (ou .NET). A sintaxe orientada a objetos foi baseada no **C++**, mas inclui muitas influências de outras linguagens de programação, como Object Pascal e Java.

**Linguagem Delphi**

Anteriormente CodeGear Delphi, Imprise Delphi e Borland Delphi, também conhecido como **Delphi**, o agora **Embarcadero Delphi** abriga um compilador, uma IDE e uma linguagem de programação; foi produzido inicialmente pela Borland Software Corporation e atualmente pela Embarcadero.

O **Delphi**, originalmente direcionado para a plataforma Windows, chegou a ser usado para o desenvolvimento de aplicações nativas para Linux e Mac OS, através do Kylix (o Kylix é um IDE para linguagens C++ e Object Pascal), e para o framework Microsoft .NET em suas versões mais recentes. O desenvolvimento do Kylix foi descontinuado.

**Linguagem Perl**

É uma linguagem de programação estável e multiplataforma, usada em aplicações de missão crítica em todos os setores, sendo destacado o seu uso no desenvolvimento web de todos os tipos. Permite a criação de programas em ambientes UNIX, MSDOS, Windows, Macintosh, OS/2 e outros sistemas operacionais. Além de ser muito utilizada em programação de formulários www e em tarefas administrativas de sistemas UNIX - em que a linguagem nasceu e se desenvolveu -, possui funções muito eficientes para manipulação de textos.

**Linguagem Python**

É uma linguagem de programação de alto nível, interpretada, imperativa, orientada a objetos, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991. Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation. Apesar de várias de suas partes possuírem padrões e especificações formais, a linguagem como um todo não é formalmente especificada. O padrão, de fato, é a implementação **CPython**.

Foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade ou expressividade. Combina uma sintaxe concisa e clara com os recursos poderosos de sua biblioteca-padrão e por módulos e frameworks desenvolvidos por terceiros.

**Linguagem PHP**

**PHP** é um acrônimo recursivo para "Hypertext Preprocessor”. É uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico na World Wide Web, como, por exemplo, a Wikipédia. Surgiu por volta de 1994, como um pacote de programas CGI criado por Rasmus Lerdorf, com o nome Personal Home Page Tools, para substituir um conjunto de scripts Perl usado em sua página pessoal. Em 1997 foi lançado o novo pacote da linguagem com o nome PHP/FI, trazendo a ferramenta Forms Interpreter, um interpretador de comandos SQL.

**Linguagem Java**

É uma linguagem de programação orientada a objeto desenvolvida na década de 1990, por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems. Diferentemente das linguagens convencionais, que são compiladas para o código nativo, a linguagem **Java** é compilada para um *bytecode*, que é executado por uma máquina virtual. A linguagem de programação **Java** é a linguagem convencional da Plataforma Java, mas não sua única linguagem.

**Linguagem UML**

Unified Modeling Language é uma linguagem de modelagem de terceira geração. A **UML** não é uma metodologia de desenvolvimento, não diz como fazer ou como projetar um sistema, mas auxilia a visualizar o desenho e a comunicação entre objetos.

Basicamente, a **UML** permite que desenvolvedores visualizem os produtos de seus trabalhos em diagramas padronizados. Com uma notação gráfica, a **UML** também especifica significados, isto é, semântica. É uma notação independente de processos, embora o RUP (Rational Unified Process) tenha sido especificamente desenvolvido utilizando a UML.

**Linguagem Smalltalk**

**Smalltalk-80**, ou simplesmente **Smalltalk**, é uma linguagem de programação orientada a objeto fortemente tipada.

Em Smalltalk tudo é objeto: os números, as classes, os métodos, os blocos de código etc. Não há tipos primitivos, ao contrário de outras linguagens orientadas a objetos: strings, números e caracteres são implementados como classes em **Smalltalk**, por isso esta linguagem é considerada puramente orientada a objetos. Tecnicamente, todo elemento de Smalltalk é um objeto de primeira ordem.

**Linguagem Lisp**

É uma família de linguagens de programação concebida por John McCarthy em 1958. Em um célebre artigo, ele mostra que é possível usar exclusivamente funções matemáticas como estrutura de dados elementares. Durante os anos de 1970 e 1980, Lisp se tornou a principal linguagem da comunidade de inteligência artificial, tendo sido pioneiro em aplicações como administração automática de armazenamento, linguagens interpretadas e programação funcional.

O seu nome vem de LISt Processing (a lista é a estrutura de dados fundamental desta linguagem). Tanto os dados como o programa são representados como listas, o que permite que a linguagem manipule o código fonte como qualquer outro tipo de dados.

**Desenvolvimento de programas**

O computador oferece as melhores condições de resolução de problemas complexos. Para resolver problemas, constroem-se programas utilizando as linguagens de programação.

É preciso, no entanto, estar alerta para o fato de que a codificação de instruções não é o único, nem pode se afirmar que seja o principal passo na resolução de problemas.

Portanto, para se construir programas não é suficiente que se conheçam linguagens de programação.

O desenvolvimento de um programa comporta cinco etapas.

* **Definição do problema**

O primeiro passo para resolver um problema é compreendê-lo e defini-lo claramente. A definição é uma descrição narrativa do problema.

* **Análise do problema**

Antes de codificar as instruções que comporão o programa, com base em determinada linguagem de programação, torna-se necessário descrever a lógica que justifica a sequência dessas instruções.

Esse passo é de máxima importância porque possibilita o entendimento do programa por outros que não o(s) programador(es) e facilita futuras alterações que venham a se fazer necessárias em funções de erros detectados ou evolução do problema tratado.

* **Codificação**

Após a perfeita explicitação da lógica do programa, vem a ação de codificar, isto é, aplicar vocabulário (repertório de instruções) e sintaxe próprios de alguma linguagem de programação para representar as instruções em condições de codificação pelo computador e consequente execução.

É, evidentemente, a parte essencial do processo, pois aí se está escrevendo o programa.

A linguagem utilizada deverá ser aquela aceita pelo ambiente, em nível de compatibilidade com os equipamentos instalados e com a cultura do pessoal envolvido.

* **Teste e Depuração**

Após a codificação, o programa é submetido a testes e se fazem adequações.

Retiram-se, então, erros ou imperfeições, o que se dá o nome de “debugar”. Esse termo originou-se quando, em 1945, em Harvard, a pesquisadora Grace Hopper descobriu que era um inseto (*bug*) que estava causando erro em seus trabalhos, alojado no interior do computador MARK II em que ele trabalhava.

* **Documentação**

Nessa etapa, todos os documentos são conferidos, acrescentadas outras informações que se fizerem necessárias e guardadas todas essas informações como subsídios futuros. Essa coletânea é extremamente valiosa para quem desejar fazer alterações no programa ou desenvolver problemas semelhantes no futuro.

**Programas compilados e programas interpretados**

Escreve-se, rotineiramente, um programa em linguagem de programação de alto nível. O computador, para executá-lo, armazena-o em sua memória sob a forma de dígitos binários (em linguagem de máquina).

Essa conversão do 1º estágio (quando o programa é chamado programa-fonte) para o 2º estágio (programa-objeto ou módulo-objeto) é feita por dois programas do sistema operacional denominados, respectivamente, compilador e montador.

Para executar um programa em linguagem de alto nível existe outro procedimento alternativo: a submissão a um interpretador, que, por trabalhar por etapas, é algo menos eficiente que o compilador.

Nesse caso, a “tradução” é feita paulatinamente: traduz-se uma instrução; executa-se essa instrução; traduz-se outra; executa-se outra…

Se houver alguma instrução errada, ela será detectada no justo momento da sua tradução (isto é, as anteriores a ela serão executadas normalmente).

Portanto, em conclusão:

* A área de memória requerida para operar com um programa que está sendo interpretado é menor que a requerida pela compilação.
* O tempo para execução interpretada é maior que o tempo para execução compilada, uma vez que todas as instruções têm que ser interpretadas em todas as oportunidades de execução do programa.

**Inteligência Artificial**

As grandes potências, particularmente o japão e os Estados Unidos, vêm investindo, maciçamente, no projeto de computadores em condições de exercer algumas faculdades por enquanto restritas ao cérebro humano: capazes de aceitar comandos de leigos e, como cérebros humanos, associar informações gravadas na memória para, em função delas, desenvolver raciocínios lógicos.

Falta ainda, no entanto, a ideia clara de associar materiais e obter o raciocínio como resultado: levar um computador a “aprender” e, portanto, a se programar sozinho para resolver problemas, desde o desenho de uma peça industrial até a escolha da melhor opção de investimento para uma empresa.

Deseja-se um computador que pense, que não se limite a processar dados, mas que faça juízos. Esbarra-se no fato de que a inteligência tem, além de poderes de lógica, altamente desenvolvidos, uma aguçada capacidade de utilizar o senso comum e de se comunicar.

**Ferramentas do escritório moderno**

As empresas, diante da oportunidade de utilizarem modernos recursos, em favor de mais eficiente administração, passaram a considerar a automação de seus escritórios como parte de seus planos diretores.

A automação de escritórios pode ser conceituada como a utilização de eficiência, com consequente crescimento de produtividade.

Entre os recursos mais poderosos postos hoje à disposição, destacam-se os programas de apoio voltados a:

* processar textos;
* construir planilhas eletrônicas;
* gerenciar base de dados;
* fazer editoração eletrônica.

**Referências**

* “*Informática: Conceitos Básicos* by Fernando Velloso. - 10. ed. - (Elsevier, 2017).