Inatel

S201 – Paradigmas de Programação

OS QUATRO PRINCIPAIS PARADIGMAS

Marcelo Vinícius Cysneiros Aragão

marcelovca90@inatel.br

TÓPICOS

Tópicos

- Introdução
- Os Quatro Principais Paradigmas
- Linguagens e Técnicas de Programação
- Integração de Técnicas de Programação
- Aprendendo Paradigmas de Programação

INTRODUÇÃO

Introdução

- Vários estilos (modelos ou paradigmas) de programação – imperativos, funcionais, lógicos e orientados a objetos – foram desenvolvidos durante mais de quarenta anos de história da programação.
- Cada um deles é baseado em **abstrações algorítmicas específicas de dados, operações e controle** e apresenta um modo específico de pensar sobre o programa e sua execução.

Introdução

- Várias **técnicas de programação** (incluindo estruturas de dados e mecanismos de controle) foram elaboradas de forma bastante independente dentro de cada estilo, formando assim diferentes **escopos de sua aplicabilidade**.
- Exemplo: o estilo OO e as técnicas correspondentes são adequados para criar programas com dados e interface complicados, enquanto o estilo lógico é conveniente para inferência lógica do programa.

Introdução

- Embora as LPs modernas geralmente incluam técnicas de programação de estilos diferentes, elas podem ser classificadas de acordo com o **estilo principal e as técnicas suportadas**.
- Por exemplo, a linguagem de programação Lisp é uma linguagem funcional, embora inclua algumas construções de programação imperativas.

Introdução

- Atualmente, para implementação de grandes projetos, são necessárias **técnicas de diferentes paradigmas**, principalmente devido à complexidade e heterogeneidade dos problemas em solução.
- Alguns deles são problemas de **processamento de dados** simbólicos complexos, para os quais as técnicas de programação de **linguagens funcionais e lógicas** (por exemplo, Lisp ou Prolog) são adequadas.
- Os outros problemas podem ser facilmente resolvidos por meio de **linguagens orientadas a objetos imperativas** populares, como C++.

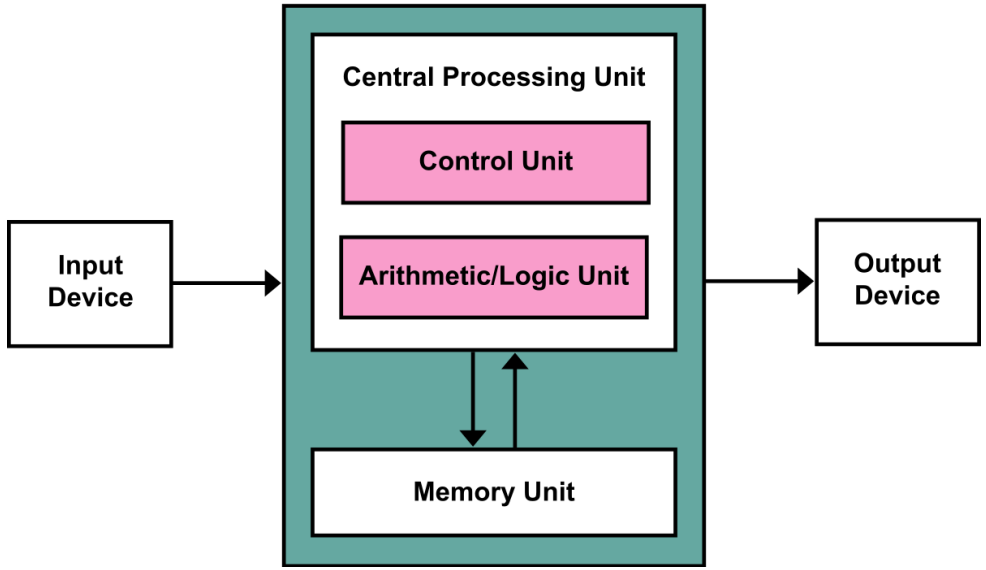
Introdução

- A seguir, será explicado o ponto de que a aquisição de técnicas de programação de todos os paradigmas principais pertence ao conhecimento prévio no campo da ciência da computação.
- Assim, o aprendizado de linguagens de programação modernas deve ser complementado e aprofundado pelo aprendizado de paradigmas de programação e suas técnicas básicas.

OS QUATRO PRINCIPAIS PARADIGMAS

Os Quatro Principais Paradigmas: **Paradigma Imperativo**

- O paradigma de programação imperativo (procedural) é o mais antigo e o mais tradicional.
- Ele cresceu a partir de **linguagens de máquina e *assembler***, cujas características principais refletem os princípios de arquitetura de computador de von Neumann.

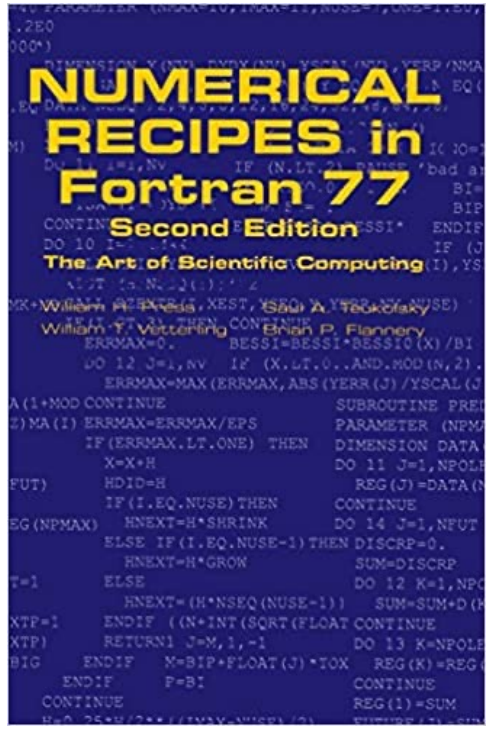


Os Quatro Principais Paradigmas: Paradigma Imperativo

- Um programa imperativo consiste em **comandos explícitos** (instruções) e **chamadas de procedimentos** (sub-rotinas) a serem executados na sequência; eles realizam **operações nos dados** e **modificam os valores das variáveis do programa** (por meio de instruções de atribuição), bem como o ambiente externo.
- Dentro desse paradigma, as variáveis são consideradas *containers* para dados semelhantes às células de memória do computador.

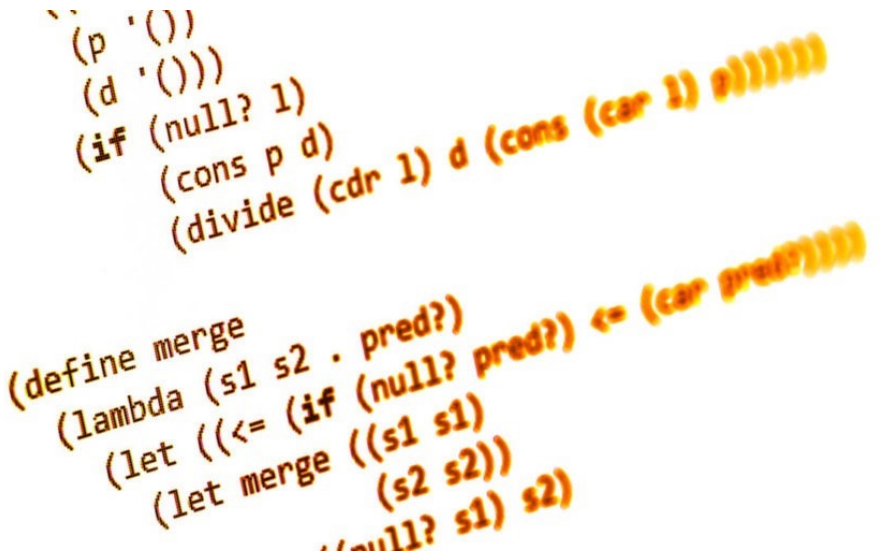
Os Quatro Principais Paradigmas: **Paradigma Funcional**

- O paradigma funcional também é um estilo antigo, uma vez que surgiu da avaliação de fórmulas algébricas, e seus elementos foram usados em linguagens algorítmicas imperativas como o Fortran.
- Programa funcional puro é uma coleção de funções mutuamente relacionadas (e possivelmente recursivas).



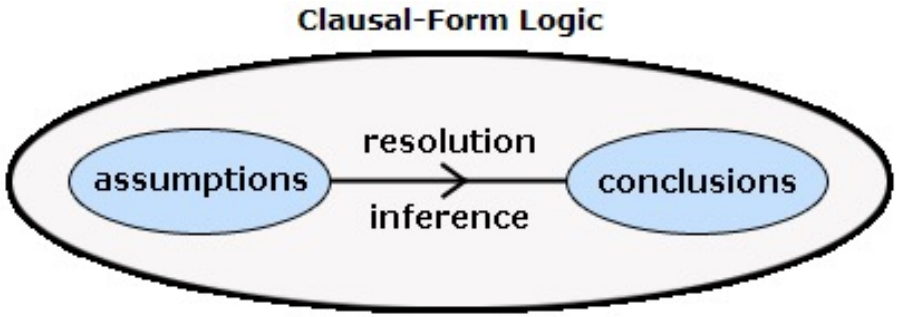
Os Quatro Principais Paradigmas: **Paradigma Funcional**

- Cada função é uma expressão para **calcular um valor** e é definida como uma composição de funções padrão (integradas).
- A execução do programa funcional é simplesmente a aplicação de todas as funções aos seus argumentos e, portanto, o cálculo de seus valores.



Os Quatro Principais Paradigmas: Paradigma Lógico

- Dentro do paradigma lógico, o programa é pensado como um conjunto de **fórmulas lógicas**: axiomas (**fatos e regras**) que descrevem propriedades de certos objetos e um teorema a ser provado.
- A execução do programa é um processo de prova lógica (**inferência**) do teorema através da construção dos objetos com as propriedades descritas.



Os Quatro Principais Paradigmas: Paradigma Lógico

- A diferença essencial entre esses três paradigmas diz respeito não apenas ao conceito **de programa e sua execução**, mas também ao conceito de **variável de programa**.
- Em contraste com os programas imperativos, **não há instruções de atribuição** explícitas **nem efeitos colaterais** em programas funcionais e lógicos puros.
- As variáveis em tal programa são semelhantes às da matemática: elas denotam **valores reais de argumentos de função** ou denotam **objetos construídos durante a inferência**.
- Essa peculiaridade explica porque os paradigmas funcionais e lógicos são considerados **não tradicionais**.

Os Quatro Principais Paradigmas: Paradigma Orientado a Objetos

- Dentro do paradigma OO, um programa descreve a estrutura e o comportamento dos chamados **objetos e classes de objetos**.
- Um objeto encapsula **dados passivos e operações ativas** nesses dados: ele tem um armazenamento que fixa seu **estado** (estrutura) e um conjunto de **métodos** (operações no armazenamento) que descrevem o comportamento do objeto.

Os Quatro Principais Paradigmas: Paradigma Orientado a Objetos

- As **classes representam conjuntos de objetos com a mesma estrutura e o mesmo comportamento**.
- Geralmente, as descrições das classes compõem uma **hierarquia** de herança, incluindo polimorfismo de operações.
- A execução de um programa orientado a objetos é considerada uma **troca de mensagens entre objetos**, modificando seus estados.

Os Quatro Principais Paradigmas

Paradigm	***Key concept***	***Program***	***Program execution***	***Result***
Imperative	Command (instruction)	Sequence of commands	Execution of commands	Final state of computer memory
Functional	Function	Collection of functions	Evaluation of functions	Value of the main function
Logic	Predicate	Logic formulas: axioms and a theorem	Logic proving of the theorem	Failure or Success of proving
Object-oriented	Object	Collection of classes of objects	Exchange of messages between the objects	Final state of the objects' states

Os Quatro Principais Paradigmas

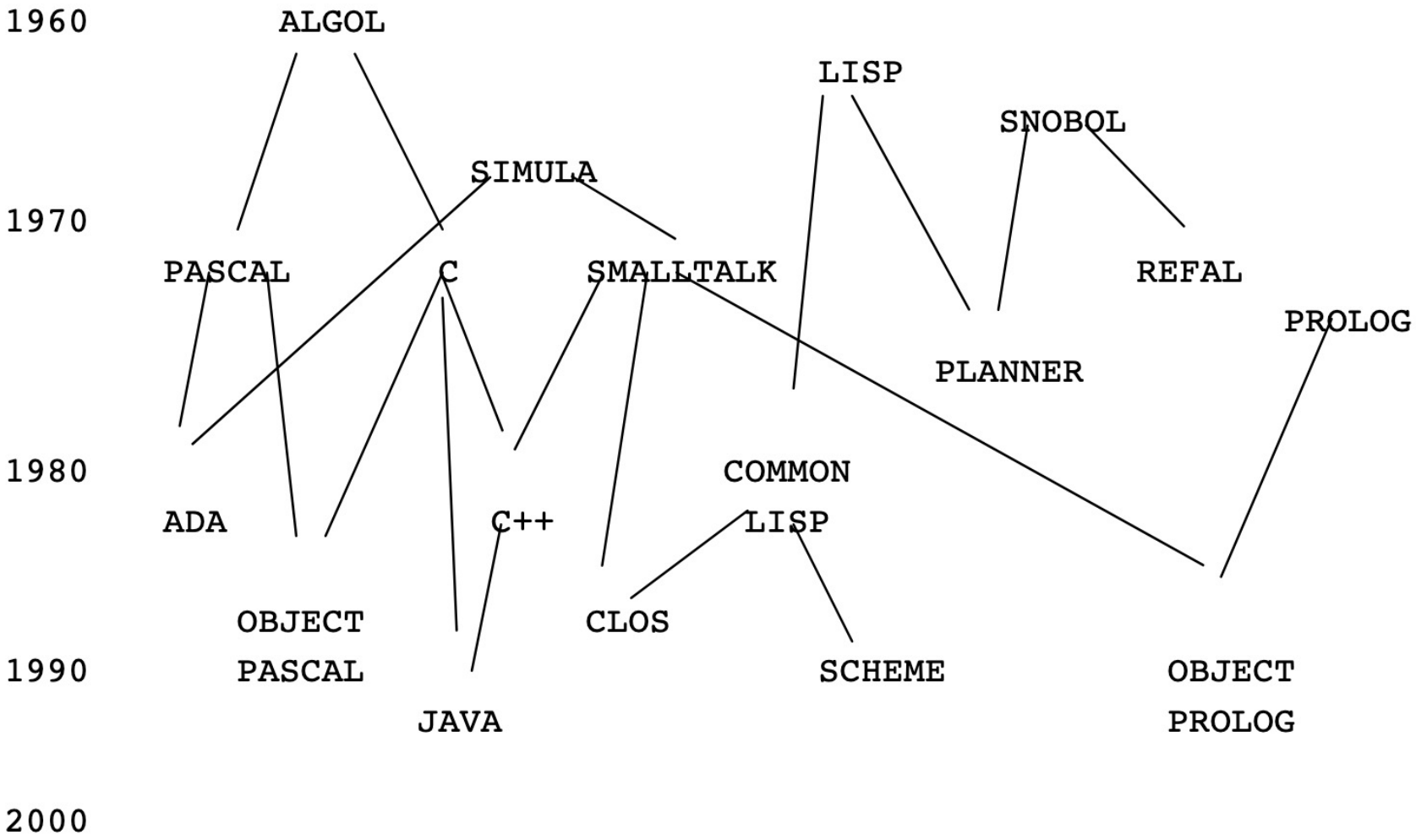
- O paradigma OO é o **mais abstrato**, pois suas ideias básicas podem ser facilmente combinadas com os princípios e técnicas dos outros estilos.
- Na verdade, um **método de objeto pode ser interpretado como um procedimento ou uma função**, enquanto o **envio de mensagem como uma chamada de procedimento ou função**.
- Contrariamente, o paradigma imperativo tradicional e os não tradicionais funcionais e lógicos são mal integrados devido a sua diferença essencial.

LINGUAGENS E TÉCNICAS DE PROGRAMAÇÃO

Linguagens e Técnicas de Programação

- Cada linguagem algorítmica foi inicialmente desenvolvida **dentro de um paradigma particular**, mas posteriormente ela geralmente acumula elementos de técnicas de programação de **outros estilos e linguagens**.
- Portanto, como regra, a maioria das linguagens inclui um núcleo que compreende **técnicas de programação de um paradigma** e também **algumas técnicas de outros paradigmas**.
- Classifica-se LPs de acordo com os paradigmas de seus núcleos.

Linguagens e Técnicas de Programação



bônus

Linguagens e Técnicas de Programação

- A seguir está uma classificação de **várias línguas famosas**, no contexto dos **principais paradigmas**:
  - Paradigma imperativo: Fortran, Algol, Pascal, Ada, C;
  - Paradigma funcional: Lisp, Scheme, Haskell, Scala, F#, Erlang;
  - Paradigma lógico: Prolog;
  - Paradigma orientado a objetos: Smalltalk, Eiffel, C#, Java.

Linguagens e Técnicas de Programação

- Smalltalk, a primeira LP orientada a objetos, não é popular devido à **complexidade de sua sintaxe** e **semântica dinâmica**.
  - Suas ideias básicas de objeto (abstração do estado e comportamento do objeto, encapsulamento e herança de estado e comportamento, polimorfismo de operações) são facilmente integradas com os princípios de LPs de outros estilos.
- Por esse motivo, o paradigma orientado a objetos se espalhou assim que foi **combinado com o paradigma imperativo tradicional**.
  - Para ser mais preciso, ele se espalhou quando foi incorporado às linguagens imperativas populares C e Pascal, dando assim as linguagens orientadas a objetos imperativas C ++ e Object Pascal.

Linguagens e Técnicas de Programação

- A **integração** análoga de princípios **orientados a objetos** com técnicas de programação de **outros paradigmas** levou a variantes orientadas a objetos de linguagens não tradicionais.
- Por exemplo, a linguagem CLOS é um **Lisp orientado a objetos** desenvolvido com base no Common Lisp, a versão popular do Lisp.
- As LPs modernas, que são combinações de dois paradigmas, são:
  - Imperativo e OO: C++, Object Pascal, Ada-95, Java;
  - Funcional e OO: CLOS;
  - Lógico e OO: Object Prolog e Logtalk.

Linguagens e Técnicas de Programação

- As técnicas de programação elaboradas dentro do paradigma imperativo tradicional e linguagens imperativas são bem conhecidas:
  - as **estruturas de controle** incluem declarações, procedimentos e funções cíclicas e condicionais, enquanto as **estruturas de dados** compreendem tipos escalares e compostos - arrays, strings, arquivos, registros, etc.
- As técnicas de programação de linguagens orientadas a objetos imperativas também incluem **tipos de objetos** e técnicas correspondentes, como procedimentos e **funções virtuais**.

Linguagens e Técnicas de Programação

- LPs baseadas em paradigmas não tradicionais fornecem tipos de dados e estruturas de controle bastante diferentes
- Estas linguagens também diferem das tradicionais no modo de execução:
  - **interpretação em vez de compilação** aplicada para linguagens orientadas a objetos imperativas e imperativas.

Linguagens e Técnicas de Programação

- Ao contrário das linguagens imperativas, **as linguagens lógicas e funcionais são geralmente recursivas e interpretativas**, e a maioria delas é orientada para o processamento simbólico.
- Além da recursão, as técnicas de programação desenvolvidas nessas linguagens incluem:
    - estruturas de dados flexíveis para representar dados simbólicos complexos, como lista (Lisp) ou termo (Prolog);
    - recursos de correspondência de padrões (Refal, Prolog) e *backtracking* automático (Planner, Prolog);
    - funcionais, i.e. funções de ordem superior (Lisp, Scheme);
    - mecanismo de avaliações parciais (Refal).

Linguagens e Técnicas de Programação

- As técnicas de programação elaboradas dentro do estilo de programação e linguagens de programação correspondentes têm seu próprio escopo de aplicações adequadas.
  - A **programação funcional** é preferível para **processamento simbólico**, enquanto a **programação lógica** é útil para bancos **de dados dedutivos e sistemas especialistas**, mas ambos não são adequados para tarefas interativas ou aplicativos de direcionamento de eventos.
  - As **linguagens imperativas** são igualmente convenientes para **cálculos numéricos e simbólicos**, abrindo mão da maioria das linguagens funcionais e do Prolog no poder das técnicas de processamento simbólico.
  - O **paradigma OO** é útil para criar **programas grandes** (especialmente interativos) com comportamento complicado e com vários tipos de dados.

INTEGRAÇÃO DE TÉCNICAS DE PROGRAMAÇÃO

Integração de Técnicas de Programação

- Hoje em dia, as **linguagens OO imperativas (como C++ e Java)**, suportadas por um grande número de ferramentas de desenvolvimento, são a escolha mais popular para implementação de **grandes projetos** de programação.
- No entanto, essas linguagens são insuficientemente adequadas para a implementação de grandes projetos de software, nos quais um ou vários problemas geralmente pertencem ao domínio do **processamento simbólico**, onde linguagens não tradicionais, como Lisp ou Prolog, são mais adequadas.

Integração de Técnicas de Programação

- Hoje em dia, as **linguagens OO imperativas (como C++ e Java)**, suportadas por um grande número de ferramentas de desenvolvimento, são a escolha mais popular para implementação de **grandes projetos** de programação.
- No entanto, essas linguagens são insuficientemente adequadas para a implementação de grandes projetos de software, nos quais um ou vários problemas geralmente pertencem ao domínio do **processamento simbólico**, onde linguagens não tradicionais, como Lisp ou Prolog, são mais adequadas.

"realizar analiticamente diferenciação, integração, simplificação, transformações e resolução de equações"

Integração de Técnicas de Programação

- Por exemplo: o desenvolvimento de um **banco de dados com uma estrutura complexa** (com centenas de relações) e com uma **interface de linguagem natural** (consultas escritas como sentenças de um subconjunto restrito de linguagem natural e respostas em uma forma de LN conveniente) envolve os seguintes problemas a serem resolvidos:

Problems	***Suitable programming languages***
Syntactic analysis of NL query	Refal
Semantic analysis of the query	Lisp, Prolog
Processing of the query	Prolog
Elaboration of response	Lisp, Refal
Modern user interface	C++, Object Pascal, Java
DB managing operations	C++

- À direita dos problemas, as LPs adequadas correspondentes são indicadas.

Integração de Técnicas de Programação

- Por exemplo: o desenvolvimento de um **banco de dados com uma estrutura complexa** (com centenas de relações) e com uma **interface de linguagem natural** (consultas escritas como sentenças de um subconjunto restrito de linguagem natural e respostas em uma forma de LN conveniente) envolve os seguintes problemas a serem resolvidos:

Problems	***Suitable programming languages***
Syntactic analysis of NL query	Refal
Semantic analysis of the query	Lisp, Prolog
Processing of the query	Prolog
Elaboration of response	Lisp, Refal
Modern user interface	C++, Object Pascal, Java
DB managing operations	C++

- LP criada na Rússia (então URSS) na segunda metade da década de 1960.
- É uma LP funcional na qual a computação é conduzida por correspondência de padrões em informações textuais e simbólicas.
- https://en.wikipedia.org/wiki/Refal

- À direita dos problemas, as LPs adequadas correspondentes são indicadas.

Integração de Técnicas de Programação

- Evidentemente, as LPs orientadas para o **processamento simbólico** são preferíveis para a análise sintática e semântica de consultas em **linguagem natural**, bem como para a geração de frases em LN que expressem respostas.
  - Mas nem sempre! https://ai.stackexchange.com/questions/27761/what-language-is-the-gpt-3-engine-written-in
- A análise semântica das consultas frequentemente implica alguma **inferência lógica**, que está disponível no **Prolog**.
- Assim, para facilitar a implementação de projetos de programação, é necessária uma integração de técnicas de programação de diferentes linguagens e estilos.
- Parece atraente **aumentar o poder das LPs OOs imperativas populares com estruturas de dados e mecanismos de controle de LPs não tradicionais**.

Integração de Técnicas de Programação

- Para resolver a necessidade de **integração de várias técnicas de programação**, que surgiu muito antes do aparecimento de linguagens OO populares, duas maneiras foram propostas:
  - A primeira forma sugere **introduzir, na implementação da LP em questão, um procedimento análogo** àquele presente na LP que apresenta tal técnica.
    - Essa maneira exige **muito trabalho e não preserva a sintaxe e a semântica** das técnicas originalmente oferecidas pela LP em questão.

Integração de Técnicas de Programação

- Outra forma de integração envolve a **codificação de cada problema em uma linguagem de programação apropriada** e a **integração dos módulos** de programa resultantes com o auxílio de um sistema operacional multitarefa (por exemplo, iniciando seu próprio processo para cada módulo).
  - Esta forma é difícil de realizar devido à **natureza fechada** de implementação de linguagens não tradicionais e **incompatibilidade de estruturas de dados** de diferentes linguagens (por exemplo, estruturas de dados de Prolog e C++ são incompatíveis).
  - https://stackoverflow.com/questions/636841/how-do-multiple-languages-interact-in-one-project/

Integração de Técnicas de Programação

- A primeira forma foi desenvolvida recentemente para integrar várias técnicas de programação com base em uma LP OO imperativa, como C++ ou Object Pascal.
- Devemos também notar que a segunda forma de integração, ou seja, **integração direta de códigos de programação escritos em diferentes linguagens**, agora se torna uma perspectiva em conexão com o desenvolvimento da plataforma Microsoft.NET, que permite compilar e vincular esses diferentes códigos.
  - https://docs.microsoft.com/pt-br/dotnet/standard/native-interop/

APRENDENDO PARADIGMAS DE PROGRAMAÇÃO

Aprendendo Paradigmas de Programação

- **A necessidade de integrar várias técnicas e LPs dentro do mesmo projeto de software é uma reivindicação da programação moderna.**
- Portanto, uma formação profunda no campo da ciência da computação deve ser baseada no aprendizado de técnicas de programação de **diferentes paradigmas**.
- Isso implica aprender várias linguagens algorítmicas diferentes, pois nenhuma das linguagens pode abranger todas as técnicas possíveis de vários estilos de programação.

Aprendendo Paradigmas de Programação

- A importância de aprender paradigmas de programação também é explicada pelo fato de que **não podemos saber o futuro das linguagens modernas populares**.
- Durante a história da programação, **muitas linguagens morreram, enquanto algumas outras perderam sua popularidade**, então algumas linguagens modernas podem ter o mesmo destino.
- No entanto, os principais paradigmas de programação serão os mesmos, bem como as suas técnicas de programação de base; logo, sua aprendizagem é dada como **permanente** na área de computação.

CONCLUSÃO

Conclusão

- Descrevemos os principais paradigmas de programação, bem como técnicas de programação e linguagens de programação elaboradas dentro deles.
- As técnicas de programação do paradigma imperativo tradicional diferem essencialmente das técnicas dos não tradicionais – funcionais e lógicas.
  - Eles têm diferentes escopos de aplicabilidade, e por esta razão a necessidade de integrar técnicas de diferentes paradigmas frequentemente surge em projetos de programação.
- Assim, uma educação profunda em ciência da computação implica a aquisição de técnicas de programação de todos os paradigmas principais, e o aprendizado usual de LPs modernas deve ser complementado pelo aprendizado de paradigmas de programação e suas técnicas de programação de base.

Referência Bibliográfica

- SEBESTA, Robert W.; SANTOS, José Carlos Barbosa dos; TORTELLO, João Eduardo Nóbrega, Conceitos de linguagens de programação. 11 ed. Porto Alegre, RS: Editora Bookman, 2018, 758 p. ISBN 978-85-8260-468-7.
- BOLSHAKOVA, Elena. Programming paradigms in computer science education. 2005.