VIDYA

VIDYA, v. 45, n. 2, p. 23-43, jul./dez. - Santa Maria, 2025.

Recebido em: 10.04.2025. Aprovado em: 30.11.2025.
ISSN 2176-4603 | DOI: doi.org/10.37781/vidya.v45i2.5341

APLICAÇÃO DA TEORIA DOS CAMPOS CONCEITUAIS NA PROGRAMAÇÃO DE COMPUTADORES ENVOLVENDO A RESOLUÇÃO DE UM PROBLEMA DE DILATAÇÃO TÉRMICA

APPLICATION OF CONCEPTUAL FIELD THEORY IN COMPUTER PROGRAMMING INVOLVING THE RESOLUTION OF A THERMAL DILATATION PROBLEM

APLICACIÓN DE LA TEORÍA DE CAMPOS CONCEPTUAL EN PROGRAMACIÓN INFORMÁTICA QUE IMPLICA LA RESOLUCIÓN DE UN PROBLEMA DE DILATACIÓN TÉRMICA

MARCOS LUÍS CASSAL[1]
SILVIA MARIA DE AGUIAR ISAIA[2]
GILBERTO ORENGO[3]

RESUMO

Este trabalho tem como objetivo apresentar a aplicabilidade da Teoria dos Campos Conceituais na programação de computadores, é discutido o desenvolvimento de uma aplicação dos fundamentos da teoria, com um exemplo empírico em um problema de dilatação térmica. A Teoria dos Campos Conceituais é uma teoria de aprendizagem cognitivista que tem como princípio a organização do conhecimento em campos conceituais. Justifica-se a importância desse tema e a carência de investigações, pois ainda são poucos os trabalhos desenvolvidos que exploram as teorias de aprendizagem no ensino da programação de computadores. É divulgada a exposição de um plano que possa ser empregado na implementação de um programa computacional. A resolução do problema envolve dois campos conceituais, o da Física e o da Programação de Computadores em Python. Os resultados demonstram que a integração da teoria de Vergnaud com a programação de computadores pode colaborar com desenvolvimento da área.

Palavras-chave: Gérard Vergnaud; TCC; Python.

ABSTRACT

This work aims to present the applicability of the Theory of Conceptual Fields in computer programming, the development of an application of the foundations of the theory is discussed, with an empirical example in a problem of thermal expansion. The Conceptual Fields Theory is a cognitive learning theory that has as its principle the organization of knowledge in conceptual fields. The importance of this topic and the lack of investigations are justified, since there are still few works developed that explore the theories of learning in the teaching of computer programming. An exposition of a plan that can be used in the implementation of a computer program is presented. Solving the problem involves two conceptual fields, Physics and Computer Programming in Python. The results demonstrate that the integration of Vergnaud's theory with computer programming can collaborate with the development of the area.

Keywords: *Gérard Vergnaud; TCC; Python.*

RESUMEN

Este trabajo tiene como objetivo presentar la aplicabilidad de la Teoría Conceptual de Campos en la programación de computadoras, se discute el desarrollo de una aplicación de los fundamentos de la teoría, con un ejemplo empírico

1 Doutor em Ensino de Ciências e Matemática. Colégio Politécnico - Universidade Federal de Santa Maria - UFSM. E-mail: cassal@politecnico.ufsm.br. ORCID: https://orcid.org/0000-0001-7326-4144.
2 Doutora em Educação. Universidade Franciscana - UFN. E-mail: silviamariaisaia@gmail.com. ORCID: https://orcid.org/0000-0002-9987-7931.
3 Doutor em Engenharia Mecânica. E-mail: g.orengo@gmail.com. ORCID: https://orcid.org/0000-0002-4846-8660

en un problema de expansión térmica. La Teoría de Campos Conceptuales es una teoría del aprendizaje cognitivista que tiene como principio la organización del conocimiento en campos conceptuales. La importancia de este tema y la falta de investigaciones se justifican, pues aún son pocos los trabajos desarrollados que exploren las teorías del aprendizaje en la enseñanza de la programación informática. Se divulga una presentación de un plan que se puede utilizar en la implementación de un programa informático. Resolver el problema involucra dos campos conceptuales, la Física y la Programación de Computadoras en Python. Los resultados demuestran que la integración de la teoría de Vergnaud con la programación informática puede contribuir al desarrollo del área.

***Palabras-clave**: Gérard Vergnaud; TCC; Python.*

INTRODUÇÃO

O número de pesquisas que tem como fundamentação teórica a Teoria dos Campos Conceituais (TCC) é relevante e se aplica a diferentes linhas, entre as quais se destaca o Ensino de Ciências (Cunha; Ferreira, 2020). A ideia que aqui está sendo apresentada é propor a aplicação desta teoria em outras áreas do conhecimento, neste caso, especificamente, à programação de computadores com um problema da física.

A competência programação de computadores exige do programador uma considerável habilidade de abstração, que consiste em uma forma diferente de pensar, ao identificar as principais propriedades e aspectos dos conceitos que são importantes para a situação problema que está sendo tratada, sem se apegar aos pontos secundários que, no momento, não vão influenciar na resolução do problema. Em outro aspecto é preciso apresentar uma solução computacional que possa ser aplicada em diversos tipos de situação problema que envolvam o campo conceitual que está sendo tratado, atendendo a necessidades do referido domínio do problema. O resultado somente será satisfatório se este programa conseguir atender as exigências de quem o utilizará.

Um software é produzido com o propósito de auxiliar a execução de uma determinada tarefa usando o computador. O programador ao se deparar com uma determinada situação problema terá que trabalhar com mais de um campo conceitual. Como o software possivelmente vai atender a necessidade de uma outra área do conhecimento, o programador também terá que conhecer o campo conceitual desta área. As situações que compõem o problema, para o qual está se propondo o desenvolvimento do software, tem as suas particularidades, segundo a TCC de Vergnaud tem seus esquemas, invariante operatórios, significados e significantes. Portanto, para que se consiga apresentar uma solução adequada, o programador terá que primeiramente aprender o campo conceitual do problema, para depois iniciar o processo de desenvolvimento do software, ocasionando a combinação do campo conceitual do problema em questão com o campo conceitual da programação.

Neste artigo será apresentado e discutido um seguimento de conteúdos que tem o objetivo de trabalhar com uma situação prática, envolvendo a solução computacional de um problema de dilatação térmica. A proposta foi elaborada com fundamentação na TCC, com a intenção de possibilitar uma reflexão de como esta teoria pode ser desenvolvida e aplicada na programação de computadores.

O artigo está organizado com a seguinte estrutura: na próxima seção é exposta a Teoria dos Campos Conceituais de Gérard Vergnaud, posteriormente é apresentada uma nova proposta envolvendo a aplicação da TCC na programação de computadores. A seção seguinte discute as estratégias empregadas com a programação de computadores, baseada na teoria de Vergnaud, na solução computacional de um problema da Física, a dilatação térmica. O trabalho encerra-se com as conclusões.

TEORIA DOS CAMPOS CONCEITUAIS

A teoria de Vergnaud foi desenvolvida com o propósito de compreender os problemas de desenvolvimento no interior de um mesmo campo de conhecimento, procurando propiciar uma estrutura às pesquisas sobre atividades cognitivas complexas, em especial com referência às aprendizagens científicas e técnicas. A ideia principal é apresentar uma teoria geral para o desenvolvimento, que procura relacionar a evolução do aprendiz com as tarefas que esse sujeito é levado a resolver. A cognição tem como cerne a conceitualização, um processo longo e que requer uma diversificação de situações.

A Teoria dos Campos Conceituais é complexa, compreende em uma única perspectiva teórica o desenvolvimento de situações, conceitos e teoremas necessários para concretizar eficientemente tais situações e dos símbolos e palavras que descrevem de maneira eficaz esses conceitos e operações (Vergnaud; Moreira, 2017).

Um conceito envolve um conjunto de situações que lhe dão significado: um conjunto de invariantes (propriedades do conceito) subjacentes ao raciocínio e um conjunto de símbolos para sua representação (Vergnaud, 2017) [p. 18].

O conceito existe porque existem situações, problemas pragmáticos e teóricos, pelos quais alguém tem interesse. É preciso estabelecer o vínculo entre a formação dos conceitos em situação na ação e, a seguir, de forma textual enunciativa que o conhecimento adquire quando está organizado em texto. Os conceitos ao mesmo tempo que orientam o enfrentamento de um problema são resultado da resolução de problemas e se organizam em forma de esquemas, isto é, uma organização invariante da conduta diante de uma classe de situações (Vergnaud, 2017).

Situações

Para Vergnaud (2017) a maioria dos conceitos são locais, as conceitualizações são moldadas pelas situações enfrentadas pelo aprendiz. Ao enfrentar uma situação é que se consegue entender a importância do esquema e da organização da conduta, que estabelece metas, regras de ação, invariantes operatórios (conhecimento que tem mais a ver com o fazer do que com o dizer) e inferências.

Um campo conceitual é tempo um conjunto de situações e um conjunto de conceitos. O conjunto de situações implica uma variedade de conceitos, de esquemas e de representações simbólicas em estreita conexão; o conjunto de conceitos que contribuem a dominar estas situações (Vergnaud; Moreira, 2017) [p. 42].

O conceito de situação não tem aqui o sentido de situação didática, mas o de tarefa a ser executada. A ideia é que toda situação complexa pode ser analisada como uma combinação de tarefas, cuja natureza e dificuldades específicas devem ser bem conhecidas. O fracasso em uma subtarefa provoca o fracasso global.

Para Vergnaud (1993), o tratamento de situações supõe, ao mesmo tempo, a identificação das questões e a das operações que devem ser executadas para resolvê-las. Toda situação, pode ser conduzida a uma combinação de relações de base com dados conhecidos e desconhecidos, que correspondem ao número de questões possíveis.

Os esquemas se adaptam, se modificam ao se depararem com novas situações, estão relacionados a organização das condutas do aprendiz para uma classe de situações, classe essa que é essencial para referenciar um esquema e seus subprodutos (procedimentos, estratégias,

regras de ação, ...), é composto de regras (implícitas ou explícitas), que são necessariamente determinadas pela representação (implícita ou explícita) das relações em cena na situação tratada (Vergnaud, 2017).

Esquema

Um esquema é uma totalidade organizada que permite gerar uma classe de comportamentos diferentes em função das características particulares de cada situação da classe a que se destina. Por meio da análise das estratégias utilizadas na referida situação, os esquemas, bem como os modelos mentais construídos frente a novas situações baseiam-se no conjunto de sentidos, pressupostos, regras de raciocínio, inferências, ... levando o aprendiz a fazer determinada interpretação, definindo como cada um percebe, pensa, sente e interage (Palmero; Moreira, 2004).

O esquema é uma organização invariante da atividade para uma dada classe de situação, formado basicamente por quatro componentes: metas, submetas e antecipações; regras de ação, de busca de informações e de controle; invariantes operatórios (conceitos-em-ação e teoremas-em-ação); possibilidades de inferência em situação (Vergnaud; Moreira, 2017).

Um esquema é uma função temporizada de argumentos, permite gerar diferentes sequências de ações e tomadas de informações, em função dos valores das variáveis de situação. A tomada da informação na leitura do enunciado, a tomada de informações físicas (medidas, por exemplo), a busca de informações em documentos (livros, quadros estatísticos, etc), a combinação adequada destas informações, são ações fundamentais para as operações que serão executadas frente à situação problema (Vergnaud, 1993).

Invariantes operatórios

Um invariante operatório é um conhecimento que tem mais a ver com fazer do que com o dizer. A construção do invariante operatório é decorrência de experiências concretas de aprendizagem a partir de outro que sabe mais. O importante é que essas situações anteriores deixem um rastro em quem aprendeu (Grossi, 2017)[p. 20].

Os invariantes operatórios (conceitos-em-ação e teoremas-em-ação) constituem a base conceitual implícita (o que está subentendido) que permite obter a informação pertinente e, a partir dela e dos objetivos a alcançar, inferir as regras de ação mais apropriadas. Os invariantes operatórios, além de serem determinantes das diferenças entre um esquema e outro, são constituintes essenciais dos campos conceituais.

Os conceitos-em-ação promovem a identificação dos objetos, propriedades, relações, classes e condições, dentre uma vasta quantidade de conceitos que estão disponíveis no repertório do aprendiz. Durante a situação, uma pequena parte destes conceitos é selecionada para cada ação, sendo que podem ser adequados ou não. Na maioria das vezes os conceitos-em-ação permanecem implícitos ao longo da ação do aprendiz.

Durante a situação o aprendiz seleciona uma pequena parte da informação e são justamente esses conceitos-em-ação que permitem a seleção da informação pertinente, além de selecionar os teoremas-em-ação necessários para avaliação das metas e submetas, das regras de ação, da busca de informações e de controle que possibilitam alcançá-las. Os conceitos-em-ação são articulados pelos teoremas-em-ação.

Os teoremas-em-ação são as proposições consideradas como verdadeiras frente à situação que está sendo enfrentada. Ao enfrentar uma situação o sujeito possui uma gama de conhecimentos que possibilitam a identificação dos objetos e as relações ajustadas, conduzindo a metas e regras apropriadas para o problema em questão. As inúmeras metas e diferentes regras de ação, busca de informações e controle, utilizadas nas classes de situações são indispensáveis para a assimilação de novas situações e acomodação dos esquemas (Vergnaud; Moreira, 2017).

É importante destacar, na TCC, que os conceitos-em-ação e os teoremas-em-ação são as suposições e propriedades que o sujeito elabora e mobiliza durante a resolução de um problema. Tais construções são provisórias, situadas e vinculadas à experiência do aprendiz - no caso deste estudo, o programador - que, ao interagir com o código e com o problema, formula hipóteses de funcionamento e verifica suas consequências e resultados na prática. É nessa relação dinâmica que se constrói o conhecimento, articulando ação, reflexão e reformulação dos esquemas cognitivos.

Representações Linguísticas e Simbólicas

Vergnaud insere-se entre os teóricos que consideram a existência de uma mediação entre os modos simbólicos de representação (os significantes) e os objetos do mundo material (a realidade). A representação não se reduz a um sistema simbólico remetendo diretamente ao mundo material, os significantes (símbolos e sinais) representam significados que são eles mesmos de ordem cognitiva e psicológica (Palmero; Moreira, 2004).

Segundo Vergnaud (2014) “Conhecimento consiste de significantes e significado, que não é formado somente de símbolos, mas também de conceitos e noções que refletem ao mesmo tempo o mundo material e a atividade do sujeito no mundo material” (p. 19).

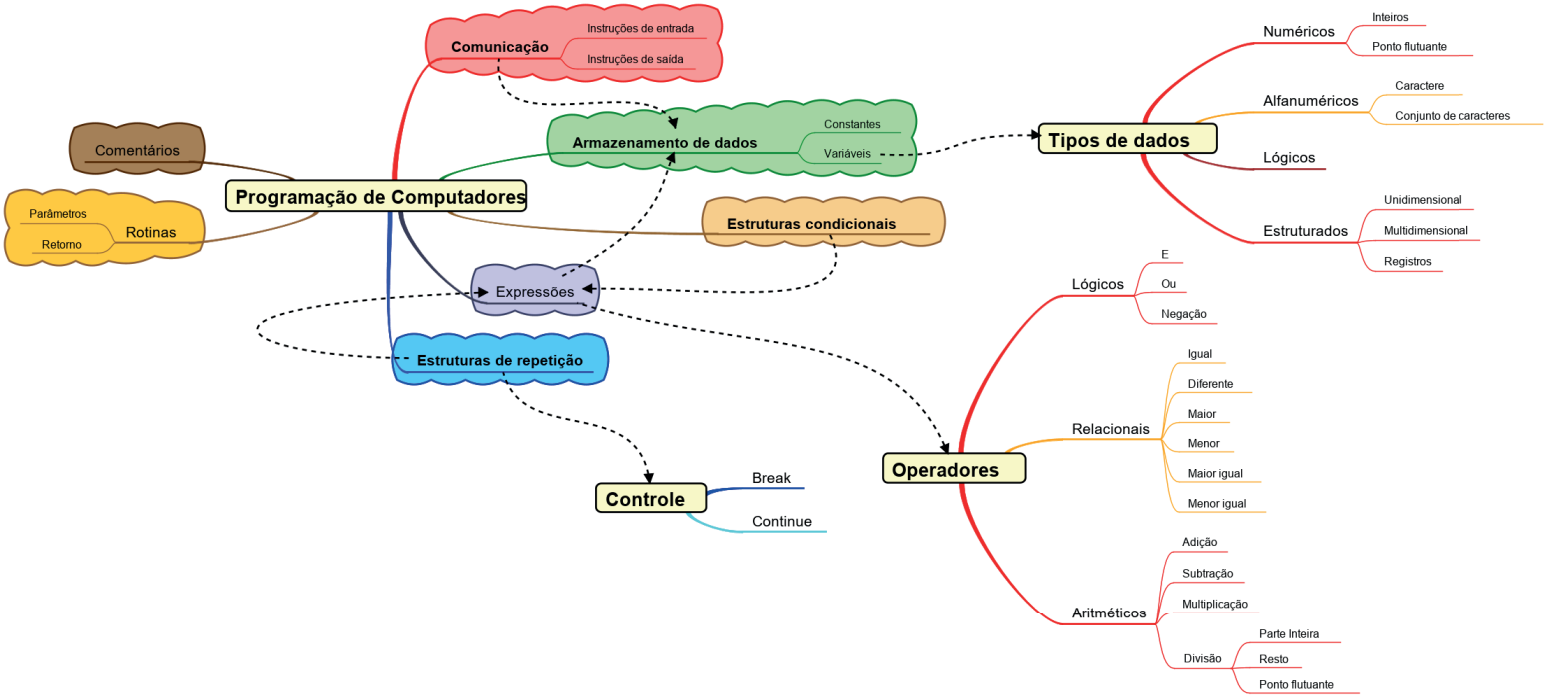
Todo instrumental de linguagem é excelente para veicular a informação, tanto na expressão da solução ou nas verbalizações que acompanham o raciocínio, quanto no próprio enunciado do problema. O simbolismo, a rigor, não é nem uma condição necessária, nem uma condição suficiente para a conceitualização. Contribui, contudo, de modo útil, para essa conceitualização (Vergnaud, 1993). A próxima seção do artigo demonstra, por meio de uma situação prática, a aplicabilidade da Teoria dos Campos Conceituais na programação de computadores.

A PROGRAMAÇÃO DE COMPUTADORES BASEADA NA TEORIA DOS CAMPOS CONCEITUAIS

Existe uma quantidade notável de linguagens de programação, cada qual com suas características próprias, com vantagens e desvantagens em comparação umas com as outras. Por este motivo é apresentado um campo conceitual que englobe conceitos que estão presentes na essência de uma linguagem de programação.

Ao desenvolver um programa de computador o programador vai se deparar basicamente com seis classes de situações: comunicação, armazenamento de dados, estruturas condicionais, estruturas de repetição, rotinas e comentários. De acordo com o tipo de situação estas classes se subdividem em subclasses e estão diretamente inseridas no raciocínio lógico aplicado na solução do problema. A Figura 1 exibe as classes de situações que compõem este campo conceitual.

Figura 1 - Teoria dos Campos Conceituais na Programação de Computadores.



Fonte: construção dos autores.

Na Teoria dos Campos Conceituais os esquemas correspondem à organização das atividades do sujeito ao se deparar com um problema, representam as hipóteses que são elencadas frente a esta classe de situações, possibilitando ao aprendiz a apropriação de um determinado campo conceitual. Programar um computador não é apenas escrever um código que será executado pela máquina. É uma atividade que envolve muitas etapas como: compreensão do problema, levantamento de dados necessário à solução, elaboração de uma sequência lógica de passos que resultará na resolução do problema, conhecer uma linguagem de programação, codificar, testar e verificar se o resultado está correto, corrigir e voltar a testar até atingir uma solução aceitável para o problema proposto. Durante a codificação, o programador elabora um algoritmo que produzirá a solução de um determinado problema. O programa se refere a uma sequência lógica de passos, escrita em uma linguagem de programação, correspondendo às instruções que serão interpretadas e executadas por um computador.

Comunicação

Um dos aspectos de importância no processo de desenvolvimento de um programa é a interatividade que deve ocorrer entre a aplicação e a pessoa que está utilizando o software, o usuário. Isso viabiliza a comunicação entre o usuário e a máquina e na maioria das vezes ocorre na forma textual. Mensagens são passadas ao usuário com solicitações de dados, informando ações que estão sendo processadas ou ainda exibindo os resultados obtidos. Esta comunicação transcorre com a utilização de instruções de leitura e escrita, responsáveis pelas operações de entrada e saída de dados.

VIDYA, v. 45, n. 2, p. 23-43,
jul./dez. - Santa Maria, 2025.

A saída de dados faz correspondência às informações que são exibidas ao usuário, geralmente fazendo uso da tela do computador. A entrada de dados está correlacionada às informações que são passadas ao software (maioria das vezes, se dá por intermédio de um teclado) e que posteriormente serão processadas.

Armazenamento de Dados

A abordagem deste conceito é em nível de armazenamento temporário de dados, isto é, a armazenagem estará ativa somente enquanto o programa estiver sendo executado. O armazenamento dos dados corresponde a um endereço da memória do computador, é um espaço que guarda um determinado valor durante a execução do programa.

Os valores que são armazenados podem ser classificados em: numéricos, alfanuméricos e lógicos. Existem outros tipos, porém esse critério está relacionado com a linguagem de programação, logo optou-se por descrever os tipos principais, que serão encontrados em quase todas as linguagens.

Existem situações que demandam a utilização de estruturas para o armazenamento de vários valores, em uma mesma variável. Esses tipos de estrutura de dados são definidos como *arrays*, variáveis indexadas, vetores ou matrizes. Um vetor possui uma única dimensão, tendo a sua organização estruturada por posição e valor.

A estrutura do tipo matriz é definida por duas ou mais dimensões. A representação com duas dimensões é a forma usual e se comparada a uma tabela, há dois índices que representam a posição de cada elemento: um identifica a linha e outro, a coluna.

Os registros são considerados tipos heterogêneos, isto é, contêm em sua estrutura outros tipos de dados. Distintos de vetores e matrizes que armazenam dados do mesmo tipo (conjuntos homogêneos).

Expressões

O uso de expressões na programação oferece a possibilidade de implementar equações matemáticas, expressões algébricas ou lógicas. Para realizar a implementação de cálculos matemáticos é necessária a utilização dos operadores, que são símbolos que determinam quais são as operações aritméticas que estão envolvidas nas expressões.

Os operadores relacionais são empregados em situações de verificação, permitindo que sejam feitos testes de comparação entre valores. Vergnaud no livro intitulado: A criança, a Matemática e a Realidade (Vergnaud, 2014), faz uma reflexão interessante sobre os operadores relacionais e as suas aplicações.

Muitos problemas exigem testes que envolvem a comparação de mais de uma condição. Nestas situações é necessário fazer uso dos operadores lógicos, para a execução de mais de um teste comparativo ao mesmo tempo. O retorno de um operador lógico é um valor booleano, ou seja, um valor verdadeiro ou falso.

Estruturas Condicionais

A execução de um código ocorre da primeira linha em direção a última. Durante esse processo existirão momentos em que o fluxo de execução poderá ter desvios, devido à lógica empregada para

definir o comportamento do programa, conforme as situações ocorrerem. O controle destes desvios por meio das estruturas condicionais.

A ação do teste condicional é baseada em uma situação que envolve uma decisão lógica, que estará apoiada em uma ou mais condições e resultará uma solução verdadeira (1) ou falsa (0). De acordo com o resultado deste teste, o programa toma a decisão de qual será o fluxo a ser seguido. A formulação de um teste, geralmente, é representada por uma expressão.

A estrutura condicional mais conhecida para a tomada de decisão é comando *if*. Ao definir um teste com o *if* , o programador faz a leitura da instrução como uma condição determinada pelo "se", isto é, "se" a condição estabelecida for verdadeira então executam-se os determinados comandos, caso contrário, os comandos não serão executados. Uma complementação ao *if* é o uso do comando *else*, quando o teste resultar em falso, outra sequência de comandos será executada.

Estruturas de Repetição

As estruturas de repetição são identificadas na literatura relacionada à programação de computadores como *loop*. O princípio básico deste tipo de estrutura está em executar um bloco de comandos "n" vezes, isto é, existe um critério de parada para o *loop*, uma condição que será testada em cada execução e que quando não for satisfeita, a repetição é encerrada, deslocando o fluxo de execução do programa para a linha seguinte ao bloco que define o laço de repetição.

As linguagens de programação oferecem diferentes formas de se trabalhar com estruturas de repetição, cada qual com suas características e aplicações. Para a construção de *loops*, comumente são encontradas nas linguagens de programação, as estruturas enquanto (*while*) e para (*for*).

Os comandos *break* e *continue* são recursos computacionais utilizados no controle das estruturas de repetição. Às vezes, o programador se depara com situações em que é necessário alterar o fluxo de execução de uma estrutura de repetição. Essas alterações podem ser realizadas desconsiderando interações, ou então, interromper a execução do *loop* e deslocar o fluxo para a linha seguinte ao bloco que define os comandos pertencentes ao laço de repetição.

O comando *continue* encerra a iteração corrente, porém continua a iterar a repetição do laço. Quando este comando é acionado, o fluxo de execução do programa é deslocado para a linha que define o cabeçalho do laço, onde a condição estabelecida é testada e se for verdadeira, o programa dará prosseguimento a execução do bloco de comandos, definido para a estrutura de repetição.

Outro comando que pode ser utilizado é o *break*. Ele irá interromper completamente a execução do *looping*, ou seja, o laço é finalizado e o fluxo de execução passa para a linha seguinte à definição do bloco de comandos do laço.

Rotinas

Uma das maneiras de definir o conceito de uma rotina é fazer uma relação com a decomposição de um problema maior em problemas menores, pensando em uma divisão com partes independentes que possam ser reutilizadas em diferentes trechos do código, inclusive em outros programas, escritos pelo próprio programador ou por outros programadores. São subprogramas, que desempenham uma determinada funcionalidade dentro de um programa maior. Esse tema também é conhecido pelo termo modularização.

VIDYA, v. 45, n. 2, p. 23-43,
jul./dez. - Santa Maria, 2025.

Uma rotina pode ou não receber parâmetros. Os parâmetros correspondem a valores atribuídos a rotina e serão processados no código, eles se comportam como variáveis definidas dentro da rotina, sendo criadas no momento em que esta é invocada e destruídas quando ela se encerra. As rotinas podem ou não retornar valores, o retorno condiz com o valor obtido pelo processamento da rotina e que será retornado para a posição do código que fez a sua chamada.

Um forte indicativo de que o programador deve escrever uma rotina é quando ele perceber que o programa, ou parte dele, está executando passos iguais ou similares com regularidade. Uma rotina tem a mesma estrutura e características de um programa, com comunicação, armazenamento de dados, expressões, estruturas condicionais e estruturas de repetição.

Esta também é uma boa prática de programação, quando se refere ao controle e gerenciamento de códigos. Isto favorece o planejamento e o controle das soluções implementadas, melhorando a legibilidade do código, reduzindo o número de linhas de código por meio da reutilização. Reutilizar um código significa ter a pretensão de evitar as falhas de programação e de simplificar a manutenção do programa.

Comentários

Os comentários são importantes para documentação pois posteriormente poderão ser consultados para o entendimento da lógica empregada na solução do problema. O comentário em um programa de computador corresponde a um texto explicativo, para registrar aspectos importantes para o entendimento do código, explicando a lógica da resolução do problema.

Geralmente o conteúdo do comentário é formado pelas partes mais críticas, aquelas que exigiram mais tempo, dedicação e que são elementos chaves na solução proposta. Não é preciso comentar cada linha do código, o programador decide onde e quando fará o comentário. Outro ponto importante para a presente pesquisa é o fato dos comentários serem fontes de informação para a análise da Teoria dos Campos Conceituais, no sentido de ser possível a identificação dos conceitos empregados na solução de um determinado problema, motivo pelo qual a classe de comentários foi inserida no campo conceitual da programação.

PROBLEMA DA DILATAÇÃO TÉRMICA

Na sequência são apresentadas as estratégias planejadas para o prosseguimento das atividades, com a exposição de um plano que possa ser empregado na implementação de um programa computacional que resolva um problema de dilatação térmica. A resolução do problema envolve dois campos conceituais, o da Física e o da Programação de Computadores. Devido a essa circunstância optou-se por organizar a explanação em duas subseções: Campo conceitual da Física para um problema de dilatação térmica e Campo conceitual da programação em Python para um problema de dilatação térmica. Em ambas as abordagens são apresentadas tabelas estruturadas com base nos fundamentos da Teoria dos Campos Conceituais. A primeira coluna aponta as situações encontradas no problema, a segunda coluna, identificada por Invariantes Operatórios, organiza os Teoremas-em--ação e os Conceitos-em-ação mencionados e as Representações simbólicas aparecem na terceira coluna.

Campo Conceitual da Física para um Problema de Dilatação Térmica

Com o propósito de esclarecer e exemplificar a aplicação da Teoria dos Campos Conceituais na Programação de Computadores, é apresentado um cenário empírico, baseado em um problema da Física, de termologia, extraído do livro Imagens da Física (Amaldi, 1995), que aborda os conteúdos em nível de Ensino Médio, com o seguinte enunciado:

Uma barra de ferro tem 1 m de comprimento na temperatura de 200ºC. Calcule seu comprimento com a variação de temperatura entre 0ºC e 1000ºC. O coeficiente de dilatação do ferro é $\alpha = 12 \times 10^{-6}\ ºC^{-1}$.

Ao analisar o problema proposto percebe-se que é importante conhecer o material que sofrerá a dilatação ou contração, pois para cada material existe um coeficiente de dilatação (identificado pela letra grega α), definido em uma tabela de referência. A temperatura corresponde à energia média de agitação térmica das partículas da barra, que quando acrescida causa uma maior agitação das moléculas, efeito que aumenta a distância média entre essas moléculas e consequentemente causa a dilatação do material. E em efeito oposto, se a temperatura for reduzida o efeito de contração térmica acontecerá. Logo, deve-se conhecer a variação de temperatura da barra para saber o seu comprimento.

A Tabela 1 foi elaborada com base na Teoria dos Campos conceituais e apresenta os elementos identificados para o problema de dilatação térmica.

Tabela 1 - Campo conceitual da Física para um problema de dilatação térmica.

Situação	Invariantes operatórios		Representação simbólica
	Conceito-em-ação	Teorema-em-ação	
Variação de temperatura na barra	Temperatura inicial	200ºC	t_i
	Temperatura final	Entre 0ºC e 1000ºC	t_f
	Variação da temperatura	Diferença entre as temperaturas final e inicial	$\Delta t = t_f - t_i$
Material	Composição da Barra	Ferro	Fe
	O coeficiente de dilatação linear	$12 \times 10^{-6}\ ºC^{-1}$	α, Tabela de referência
Variação do comprimento na barra	Comprimento inicial	1 m	l_i
	Variação da temperatura	Diferença entre as temperaturas final e inicial	Δt
	Comprimento final	Variação da temperatura e coeficiente de dilatação	$l_f = l_i + \alpha l_i \Delta t$
	Contração térmica	Redução da temperatura Aumento da temperatura	$l_f = l_i(1 + \alpha \Delta t)$
	Dilatação térmica		$\Delta_l = l_f - l_i$

Fonte: construção dos autores.

A variação no comprimento pode ser calculada, para isso faz-se uso da equação

$$\Delta l = \alpha l_i \Delta t \quad (1)$$

que possibilita encontrar o comprimento final após uma Δt

$$l_f = l_i + \alpha l_i \Delta t \quad (2)$$

, na qual:

l_f = Comprimento final a ser calculado
l_i = Comprimento inicial da barra
α = Coeficiente de dilatação de acordo com o material de composição
Δt = Variação de temperatura da barra em que t_f é a temperatura final e t_i a inicial

A equação pode ser fatorada, chegando-se a:

$$l_f = l_i(1 + \alpha \Delta t) \quad (3)$$

Aplicando a equação 3 para os dados extraídos do exemplo apresentado, obtêm-se o seguinte desenvolvimento:

Temp inicial = 200ºC - Temp final = 0ºC - Comprimento inicial = 1 m
$l_f = l_i(1 + \alpha \Delta t)$
$l_f = 1(1 + 0{,}000012(0 - 200))$
$l_f = 1(1 - 0{,}0024)$
$l_f = 0{,}9976$
Comprimento final = 0,9976 m (contração)
Temp inicial = 200ºC - Temp final = 1000ºC - Comprimento inicial = 1 m
$l_f = l_i(1 + \alpha \Delta t)$
$l_f = 1(1 + 0{,}000012(1000 - 200))$
$l_f = 1(1 + 0{,}0096)$
$l_f = 1{,}0096$
Comprimento final = 1,0096 m (dilatação)

Baseando-se nos cálculos apresentados pode-se afirmar que a barra a uma temperatura de 0ºC tem o comprimento de 0,9976 m. Segundo os dados informados no próprio enunciado tem se o comprimento da barra de 1 m a uma temperatura de 200ºC. O tamanho de 1,0096 m foi calculado levando-se em consideração a temperatura de 1000ºC. Ao analisar os resultados é possível comprovar que o aumento de temperatura influencia na dilatação de materiais, aumentando também o seu comprimento. E, a redução da temperatura promoverá uma contração térmica. Percebe-se que a variação de comprimento da barra depende da quantidade de massa, além do seu material de composição. A barra sofre com a variação da temperatura, em uma única dimensão (comprimento), logo ela só poderá ser medida em materiais sólidos.

Para facilitar a assimilação da TCC com o código do programa, orienta-se o leitor a fazer uma associação das abordagens com as codificações apresentadas no corpo do texto. A numeração das linhas de cada trecho de código que será apresentada na sequência, corresponde à mesma numeração do código fonte.

A Tabela 2 representa a situação Comunicação entre o usuário e o programa segundo a TCC.

Tabela 2 - Campo conceitual da programação em Python para um problema de dilatação térmica (1).

Situação	Invariantes operatórios		Representação simbólica
	Conceito-em-ação	Teorema-em-ação	
Comunicação entre o usuário e o programa	Mensagens textuais	*print*()	Linhas: 26, 65, 92, 95, 101, 117, 121, 123, 125

Fonte: construção dos autores.

A comunicação entre usuário e a máquina se dá por intermédio da interatividade. Neste programa a comunicação ocorrerá por meio de mensagens textuais (conceito-em-ação), desta maneira o usuário receberá as orientações necessárias para que seja possível a utilização do software. O teorema-em-ação nesta situação se deu por meio da instrução (*print*()), comando da linguagem Python, que executa operações de escrita, exibindo mensagens na tela do computador. O programa realiza o processamento dos cálculos depois que forem informados o Material de composição da barra, o Coeficiente de dilatação, o Comprimento inicial da barra, Temperaturas inicial e final, como pode ser observado na Figura 2.

Figura 2 - Dados de entrada do programa.

```
Cálculo da Dilatação Linear de um Sólido
Informe o material de composição da barra: Ferro
Deseja visualizar a tabela de coeficintes lineares? (S/N)S
Informe o coeficiente de dilatação linear do(a) Ferro: 12
Informe o comprimento inicial da barra (m): 1
Informe a temperatura inicial da variação (°C): 200
Informe a temperatura final da variação (°C): 1000
```

Fonte: construção dos autores.

Os dados de entrada para o problema da dilatação térmica foram implementados em duas funções: uma que é responsável pelos valores digitados e outra que apresenta na tela os valores de referência para o coeficiente de dilatação dos materiais. A próxima tabela foi constituída pelos elementos da TCC para esta situação.

Tabela 3 - Campo conceitual da programação em Python para um problema de dilatação térmica (2).

Situação	Invariantes operatórios		Representação simbólica
	Conceito-em-ação	Teorema-em-ação	
Dados de entrada		*def entrada*	Linha 19
	Rotina	msg e tipo	Linha 19
	Parâmetros	*while*	Linha 21
	Laço de repetição	*input*()	Linha 22
	Operação de entrada	*try...except*	Linhas: 23 a 26
	Tratamento de exceção		
	Conversão entre tipos	*float()*	Linha 24
	Retorno		
		Valor convertido	Linha 29
		def tabela_referencia	Linha 81
	Rotina	Arquivo texto	Linha 84
	Armazenamento físico	*for*	Linhas: 87 e 93
	Laço de repetição	*list referencia*	Linha 90
	Estrutura de dados composta		
	Tipos de dados	*string*	Linha 103
		float	Linhas: 106 a 109

Fonte: construção dos autores.

A identificação do material de composição da barra será um texto, indicando que o tipo de dado (conceito-em-ação) para esse campo é *string* (teorema-em-ação). Os valores correspondentes ao comprimento inicial da barra, assim como o coeficiente de dilatação do material e as variações de temperatura (inicial e final), podem receber valores com parte fracionária, logo, foram definidos com o tipo *float* (teorema-em-ação). No Código 1 é possível fazer a verificação destes invariantes operatórios.

Código 1 - Implementação dos dados de entrada.

```
103 material = entrada("Informe o material de composição da barra: ")
104 if continuar("Deseja visualizar a tabela de coeficientes lineares? (S/N)"):
105     tabela_referencia()
106 alfa = entrada("Informe o coeficiente de dilatação linear do(a) " + material + ": ", float)
107 li = entrada("Informe o comprimento inicial da barra (m): ", float)
108 ti = entrada("Informe a temperatura inicial da variação (°C): ", float)
109 tf = entrada("Informe a temperatura final da variação (°C): ", float)
```

Fonte: construção dos autores.

Os valores dos dados de entrada são auferidos e validados em uma rotina (conceito-em-ação), que tem o propósito de receber os valores digitados e verificar se estes correspondem aos tipos de dados (conceito-em-ação) definidos pelo programa. A linguagem Python oferece um recurso propício para a entrada de dados, por meio da função *input*(). Quando esta função é invocada, o fluxo de execução do programa fica aguardando a digitação de um valor, até que ocorra a confirmação do valor de entrada, quando a tecla enter é pressionada. O valor informado é lido como uma *string* (teorema-em-ação), logo, quando forem necessários outros tipos dados, será necessária a conversão entre tipos (conceito-em-ação). Na função *input()*, pode-se fazer a

passagem de um texto como parâmetro, este texto será exibido na tela do computador, passando orientações ao usuário.

A validação foi implementada mediante um tratamento de exceção (conceito-em-ação) com o uso do bloco *try...except* (teorema-em-ação). Nos casos em que o dado informado estiver de acordo com a definição do tipo, uma variável irá armazenar este valor, nas outras situações, uma mensagem será exibida, alertando que o valor não condiz com o tipo de informação que deveria ser informada neste campo, além de solicitar um novo valor. Essa funcionalidade foi utilizada mais de uma vez dentro do programa, logo, optou-se por fazer a sua implementação em uma função (entrada), que recebe como parâmetros a mensagem que será exibida na tela e o tipo de dado que será utilizado na validação do valor. O retorno da função corresponde ao valor informado já convertido para o tipo de dado correspondente a sua definição, além de garantir que o valor passou por um processo de validação. Esses teoremas-em-ação podem ser visualizados no Código 2.

Código 2 - Função que valida os dados de entrada.

```
19 def entrada(msg, tipo=str):
20     '''Função que faz a validação do dado de entrada, seguindo como critério o tipo de dado que
           será armazenado na variável'''
21     while True:
22         retorno = input(msg).strip()
23         try:
24             retorno = tipo(retorno)
25         except ValueError as e:
26             print("Entrada de dados incorreta. Informe novamente")
27         else:
28             break
29     return retorno
```

Fonte: construção dos autores.

Ao solicitar o valor do coeficiente de dilatação, o programa oferece ao usuário a possibilidade de listar valores de referência para o coeficiente de dilatação. Neste ponto identifica-se um teorema-em-ação, a definição da função *tabela_referencia*, uma rotina (conceito-em-ação) que relaciona alguns materiais e seus respectivos coeficientes de dilatação. Os valores da tabela estão armazenados fisicamente (conceito-em-ação) em um arquivo texto (teorema-em-ação), no qual cada linha corresponde a um material e seu respectivo coeficiente. Para inserir novos materiais é necessário fazer a inclusão de uma nova linha no arquivo, informando o nome do material e o valor do coeficiente, separando-os com um ponto e vírgula (;). O programa faz a leitura destas informações do arquivo, as armazena em uma estrutura de dados composta (conceito-em-ação), representada no código pela lista *referencia* (teorema-em-ação), que posteriormente é percorrida com a aplicação de um laço de repetição (conceito-em-ação), do tipo *for* (teorema-em-ação), apresentando os valores na tela para que sejam consultados. No Código 3 encontra-se a implementação deste recurso.

Código 3 - Função que lista materiais com seus coeficientes de dilatação.

```
81  def tabela_referencia():
82      '''Função que lê os coeficientes dos materiais de um arquivo físico (coeficiente.txt) e os
            apresenta na forma de uma tabela'''
83      referencia = []
84      arq_coeficientes = open("coeficiente.txt", "r")
85      linha = arq_coeficientes.readlines()
86      arq_coeficientes.close()
87      for coeficiente in linha:
88          coeficiente = coeficiente.rstrip()
89          dados = coeficiente.split(";")
90          referencia.append(dados)
91      referencia.sort()
92      print("Material" + 5 * ' ' + "Coeficiente")
93      for elemento in referencia:
94          tam = 13 - len(elemento[0])
95          print(elemento[0] + tam * ' ' + elemento[1])
```

Fonte: construção dos autores.

Os invariantes operatórios que serão apresentados na próxima tabela evidenciam o cálculo da alternância do comprimento da barra, representados por duas situações.

Tabela 4 - Campo conceitual da programação em Python para um problema de dilatação térmica (3).

Situação	Invariantes operatórios		Representação simbólica
	Conceito-em-ação	Teorema-em-ação	
Variação da temperatura	Rotina	*def calcula_delta_t*	Linha 32
	Parâmetros	ti e tf	Linha 32
	Equação	Cálculo da variação da temperatura	Linha 34
	Retorno	Variação da temperatura	Linha 34
Variação do comprimento	Rotina	*def variacao_comprimento* alfa, delta_t e ti	Linha 36
	Parâmetros	Cálculo da variação de comprimento da barra	Linha 36
	Equação	Variação do comprimento	Linha 38
	Retorno		Linha 38

Fonte: construção dos autores.

A função *calcula_delta_t* (teorema-em-ação) realiza o cálculo da variação da temperatura, recebendo como parâmetros as temperaturas inicial e final e retornando (conceito-em-ação) o Δt, como pode-se conferir no Código 4. Outro teorema-em-ação representa a função que calcula a variação do comprimento (*variacao_comprimento*), recebendo como parâmetros o coeficiente de dilatação, a variação da temperatura (Δt) e o comprimento inicial da barra. O retorno da função corresponde ao comprimento final da barra. Segundo a tabela dos coeficientes de dilatação, o valor de referência dos materiais é na base 10 elevado no expoente -6, determinando que o valor informado como coeficiente de dilatação deva ser calculado, segundo esta definição.

Na linha 38 do Código 4 encontra-se o retorno da função (conceito-em-ação), representado pela equação (conceito-em-ação) que calcula a variação de comprimento da barra.

Código 4 - Funções que calculam as variações da temperatura e do comprimento.

```
32 def calcula_delta_t(ti, tf):
33     '''Função que calcula a variação do comprimento de uma barra, recebe como par metros o
           coeficiente de dilatação do material, a variação de temperatura e o comprimento inicial
           da barra'''
34     return tf - ti
35
36 def variacao_comprimento(alfa, delta_t, li):
37     '''Função que calcula a variação do comprimento de uma barra, recebe como par metros o
           coeficiente de dilatação do material, a variação de temperatura e o comprimento inicial
           da barra'''
38     return li * (1 + (alfa * (10 ** -6)) * delta_t)
```

Fonte: construção dos autores.

Na Tabela 5 estão identificados os invariantes operatórios correspondentes às operações que calculam as oscilações no comprimento da barra.

Tabela 5 - Campo conceitual da programação em Python para um problema de dilatação térmica (4).

Situação	Invariantes operatórios		Representação simbólica
	Conceito-em-ação	Teorema-em-ação	
Mudança no comprimento da barra	Comprimento final da barra	Equações	Linha 110
	Tomada de decisão	Cálculo da variação do tamanho da barra	Linha 111
		If	Linha 115

Fonte: construção dos autores.

Conhecendo-se os comprimentos inicial e final da barra é possível calcular o Δl, fazendo a subtração entre o tamanho atual e o tamanho inicial, o resultado encontrado corresponde ao quanto a barra dilatou ou contraiu. Esse valor é testado, indicando uma tomada de decisão (conceito-em-ação), uma vez que a instrução *if* (teorema-em-ação) fará a comparação do valor do (Δl), com a intenção de se certificar se o efeito ocorrido foi de dilatação ou de contração. Caso o valor do Δl seja menor do que 0 (zero), têm-se a indicativa de que ocorreu uma "contração". Valores maiores que 0 (zero), indicam que o efeito de "dilatação". Essa tomada de decisão pode ser conferida no Código 5.

Código 5 - Variação do comprimento.

```
110 lf = variacao_comprimento(alfa, calcula_delta_t(ti, tf), li)
111 delta_l = lf - li
112 temp = []
113 varia = []
114 print()
115 if delta_l < 0:
116         tipo = "Contração"
117     print("A variação do comprimento se deu em uma contração de %g (m)" % delta_l)
118     contrai(ti, tf, alfa, li)
119 else:
120         tipo = "Dilatação"
121     print("A variação do comprimento se deu em uma dilatação de %g (m)" % delta_l)
122     dilata(ti, tf, alfa, li)
123 print("Comprimento final da barra = %g (m)" % lf)
```

Fonte: construção dos autores.

Como são dois tipos de variações de tamanho que o problema menciona foram identificadas duas situações, uma para a dilatação e outra para a contração. Estas situações estão expostas na Tabela 6.

Tabela 6 - Campo conceitual da programação em Python para um problema de dilatação térmica (5).

Situação	Invariantes operatórios		Representação simbólica
	Conceito-em-ação	Teorema-em-ação	
Dilatação	Rotina	*def dilata*	Linha 42
	Parâmetros	ti, tf, alfa e li	Linha 42
	Laço de repetição	*while*	Linha 45
	Variação do comprimento	Chamada da função *variacao_comprimento*	Linha 46
		list temp e *varia*	
	Estrutura de dados composta		Linhas: 47 e 48
Contração	Rotina	*def contrai*	Linha 52
	Parâmetros	ti, tf, alfa e li	Linha 52
	Laço de repetição	*while*	Linha 55
	Variação do comprimento	Chamada da função *variacao_comprimento*	Linha 56
		list temp e *varia*	
	Estrutura de dados composta		Linhas: 57 e 58

Fonte: construção dos autores.

Como o enunciado menciona a variação de temperatura, determinada por um intervalo informado, os valores devem ser calculados “n” vezes, indicando que a implementação faça uso de uma estrutura de repetição (conceito-em-ação), variando a temperatura inicial até a final, com acréscimo ou decréscimo de 1ºC a cada iteração. No Código 6 pode-se conferir a efetivação de duas rotinas, em ambas é possível identificar a aplicação do comando *while* (teorema-em-ação) na implementação do laço de repetição, no qual encontra-se a chamada da função que calcula a variação do comprimento da barra. Uma vez calculada essa variação, são armazenadas em estruturas de dados compostas (conceito-em-ação), do tipo lista (teorema-em-ação), a temperatura atual (lista *temp*) e a variação do comprimento (lista varia).

Código 6 - Funções que calculam os efeitos de contração ou dilatação.

```
42 def dilata(ti, tf, alfa, li):
43     '''Funcao que calcula a dilatação. Recebe como par metros as temperaturas inicial e final, o
           coeficiente de dilatação e o comprimento inicial da barra'''
44     temp_atual = ti
45     while temp_atual <= tf:
46                 lf = variacao_comprimento(alfa, calcula_delta_t(ti, temp_atual), li)
47                 temp.append(temp_atual)
48                 varia.append(lf)
49         temp_atual += 1
50
51
52 def contrai(ti, tf, alfa, li):
53     '''Função que calcula a contracao. Recebe como par metros as temperaturas inicial e final, o
           coeficiente de dilatação e o comprimento inicial da barra'''
54     temp_atual = ti
55     while temp_atual >= tf:
56         lf = variacao_comprimento(alfa, calcula_delta_t(ti, temp_atual), li)
57                 temp.append(temp_atual)
58                 varia.append(lf)
59         temp_atual -= 1
```

Fonte: construção dos autores.

Na próxima tabela apresenta-se a situação correspondente a apresentação dos resultados.

Tabela 7 - Campo conceitual da programação em Python para um problema de dilatação térmica (6).

Situação	Invariantes operatórios		Representação simbólica
	Conceito-em-ação	Teorema-em-ação	
Apresentação dos resultados	Rotina Laço de repetição Estrutura de dados composta	*def mostra_dados* *for* *list temp* e *varia*	Linha 62 Linha 64 Linha 65

Fonte: construção dos autores.

O resultado apresenta o valor da variação do comprimento e o tipo de variação (dilatação ou contração), além do comprimento final. Também é exibida uma listagem com os valores correspondentes a temperatura e o respectivo comprimento, conforme pode-se observar na Figura 3.

As alterações de tamanho calculadas e as temperaturas foram armazenadas em duas estruturas de dados compostas (conceito-em-ação) do tipo lista (teorema-em-ação), uma para armazenar a temperatura e outra a variação. A apresentação destes resultados foi implementada na função *mostra_dados* (teorema-em-ação), fazendo-se uso da estrutura de repetição (conceito-em-ação) for (teorema-em-ação), percorrendo as listas que armazenam as temperaturas e as variações do comprimento. De acordo como as listas são percorridas, os valores correspondentes a temperatura e aos comprimentos correlatos são exibidos da tela. O Código 7 apresenta essa funcionalidade do programa.

Figura 3 - Variação do comprimento em função da temperatura.

```
Variação do comprimento da barra (m) em função da temperatura (°C):
Temperatura = 200 - Comprimento = 1
Temperatura = 201 - Comprimento = 1.00001
Temperatura = 202 - Comprimento = 1.00002
     .
     .
     .
Temperatura = 998 - Comprimento = 1.00958
Temperatura = 999 - Comprimento = 1.00959
Temperatura = 1000 - Comprimento = 1.0096
Deseja visualizar o gráfico? (S/N)
```

Fonte: construção dos autores.

Código 7 - Função que apresenta os dados na tela.

```
62 def mostra_dados():
63     '''Exibe a variação de temperatura e os respectivos valores calculados para a variação de
           comprimento da barra'''
64     for i in range(len(temp)):
65         print("Temperatura = %g - Comprimento = %g" % (temp[i], varia[i]))
```

Fonte: construção dos autores.

O fato de o enunciado do programa solicitar a geração de um gráfico para exibir a dilatação ou contração em consequência da temperatura, indicou a definição da situação “Representação gráfica”, apresentada na Tabela 8.

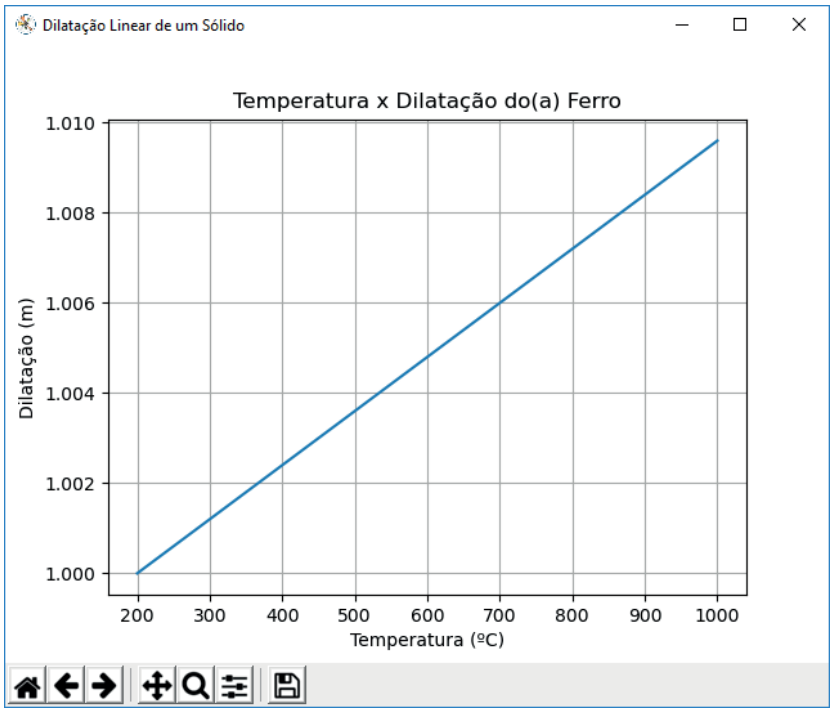
Tabela 8 - Campo conceitual da programação em Python para um problema de dilatação térmica (7).

Situação	Invariantes operatórios		Representação simbólica
	Conceito-em-ação	Teorema-em-ação	
Representação gráfica	Rotina	*def grafico*	Linha 68
	Parâmetros	material, temp, dilat e tipo	Linha 68
	Biblioteca auxiliar	*matplotlib* *pyplot*	Linha 70
	Parêmetros de configuração		Linhas: 73 a 77
	Estrutura de dados composta	*list temp* e *dilat*	Linha 77

Fonte: construção dos autores.

O programa oferece ao usuário a possibilidade de visualizar o gráfico desta relação, como o apresentado na Figura 4. Um questionamento é feito ao usuário, indagando se ele deseja visualizar o gráfico. Quando a resposta for afirmativa a geração do gráfico se dará com o auxílio da matplotlib (teorema-em-ação), uma biblioteca (conceito-em-ação) que oferece o recurso de geração de gráficos, bem como um ambiente para visualização do gráfico gerado.

Figura 4 - Representação gráfica dos valores do efeito.



Fonte: construção dos autores.

VIDYA, v. 45, n. 2, p. 23-43,
jul./dez. - Santa Maria, 2025.

Dentre os parâmetros de configuração do *pyplot* definiram-se os títulos da janela e do gráfico, a identificação das abcissas (Temperatura) e ordenadas (Dilatação ou Contração), uma grade no gráfico e a plotagem dos pontos, de acordo com os valores armazenados nas duas listas. A geração do gráfico foi implementada na função grafico (teorema-em-ação), que recebe como parâmetros o material de composição da barra, uma lista com os valores das variações de temperatura, outra lista com os valores calculados para as variações de comprimento na barra e o tipo de variação: contração ou dilatação. No Código 8 encontra-se a implementação desta situação.

Código 8 - Função que apresenta a representação gráfica do efeito.

```
68  def grafico(material, temp, dilat, tipo):
69      ''' Gera e exibe o gráfico da relação temperatura x dilatação/contração'''
70      import matplotlib.pyplot as plt
71      frm_grafico = plt.figure(0)
72      frm_grafico.canvas.set_window_title("Dilatação Linear de um Sólido")
73      plt.title("Temperatura x " + tipo + " do(a) " + material)
74      plt.xlabel("Temperatura")
75      plt.ylabel(tipo)
76      plt.grid(True)
77      plt.plot(temp, dilat)
78      plt.show()
```

Fonte: construção dos autores.

Esta seção apontou uma solução computacional para um problema de termologia, na qual o desfecho apresentado mostra a aplicação da Teoria dos Campos Conceituais, circundando um problema da Física com a área da programação de computadores. Essa conexão contempla especificidades, esquemas, invariantes operatórios, significados e significantes que ajudam no desenvolvimento cognitivo do programador.

A partir dessa perspectiva, o programador deve ser compreendido como sujeito do processo de aprendizagem, e não apenas como executor de instruções. Cada decisão tomada na elaboração de um algoritmo, expressa uma mobilização de conceitos e teoremas-em-ação. Esses invariantes operatórios são resultado da experiência, da interação com o problema e da interpretação pessoal dos campos conceituais envolvidos. Dessa forma, o código escrito materializa não apenas uma solução técnica, mas também o percurso cognitivo do sujeito que o concebeu. Ao analisar o programa sob o enfoque da TCC, é possível compreender como o programador constrói e reconstrói significados, atribuindo sentido às estruturas lógicas e simbólicas que compõem o software.

CONCLUSÕES

Este trabalho teve o propósito de descrever e aplicar a Teoria dos Campos Conceituais na área da programação de computadores. A TCC é uma teoria muito conhecida, desenvolvida e aplicada nas áreas da Matemática e da Física, não se tendo conhecimento da sua aplicação na programação de computadores, motivo inspirador para a realização desta pesquisa. A articulação dos campos conceituais da programação de computadores, com outra área do conhecimento, nesta investigação a resolução de um problema de dilatação térmica, tem um enfoque conveniente, principalmente com os conceitos do sujeito, que se encontram em processo de construção.

O estudo reforça a importância de considerar o programador como sujeito que aprende e constrói conhecimento ao programar. Ao desenvolver um software o programador vai se deparar com diferentes situações problema, derivadas do campo conceitual da programação e também do campo conceitual da área para a qual o programa está sendo construído. A aplicação da Teoria dos Campos Conceituais evidencia que a programação não se restringe à manipulação de comandos de uma linguagem de programação, mas constitui uma atividade cognitiva complexa, na qual o sujeito mobiliza, reformula e consolida invariantes operatórios. Essa articulação entre teoria e prática amplia a compreensão sobre os processos de aprendizagem, revelando que a elaboração de um software é um processo de desenvolvimento intelectual, sendo o programador um agente ativo da construção do conhecimento.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

REFERÊNCIAS

AMALDI, U. **Imagens da física: As ideias e as experiências do pêndulo aos quarks** 1st ed., 1995.

CUNHA, K. M. A., FERREIRA, L. N. d. A. **A teoria dos campos conceituais e o ensino de ciências: Uma revisão.** Revista Brasileira de Pesquisa em Educação em Ciências, 20(u), 523-552, 2020.

GROSSI, E. P. **A teoria dos campos conceituais é algo extraordinário!**, 2017.

PALMERO, M. L. R.; MOREIRA, M. A. **La teoría de los campos conceptuales de gérard vergnaud**. In M. A. Moreira (Ed.), La teoría de los campos conceptuales de vergnaud: La enseñanza de las ciencias y la investigación en el área, p. 7-39, 2004.

VERGNAUD, G. **Teoria dos campos conceituais**. In Anais do 1º seminário internacional de educação matemática do rio de janeiro, p. 1-26, 1993.

VERGNAUD, G. **A criança, a matemática e a realidade: Problemas do ensino da matemática na escola**, 2014.

VERGNAUD, G. **Piaget e vygotski em Gérard Vergnaud.** E. P. Grossi, Ed., 2017.

VERGNAUD, G.; MOREIRA, M. A. **O que é aprender? Iceberg da conceitualização** E. P. Grossi, Ed., 2017.