Modelagem de uma ontologia para construção de algoritmos com o uso da ferramenta *Protégé*

Matheus Gomes Terra[1], Ana Paula Canal[1]

[1]Ciência da Computação – Centro Universitário Franciscano
Santa Maria – RS – Brasil

matheusterra@outlook.com, apc@unifra.br

Abstract. *With the complexity of programming languages, beginner level students in computer programming are facing difficulties and for this reason, systems that help in the comprehension of these contents are useful tools. Ontologies are the representation of concepts, which through consultations deliver relevant information to the user. In this sense, the present study aims to present a modeling of an ontology based on the subjects about algorithms with the Protégé tool for the construction and development of algorithms in the C programming language.*

Resumo. *Com a complexidade das linguagens de programação, estudantes iniciantes na programação de computadores enfrentam dificuldades e, por isso, sistemas que auxiliem na compreensão desses conteúdos são ferramentas úteis. As ontologias fazem a representação dos conceitos, as quais, por meio de consultas, entregam informações relevantes ao usuário. Nesse sentido, este estudo tem como objetivo apresentar uma modelagem de uma ontologia, baseada nas disciplinas de algoritmos, com a ferramenta Protégé, para a construção e o desenvolvimento de algoritmos na linguagem de programação C.*

1. Introdução

A programação de computadores é uma atividade altamente complexa, que envolve subtarefas ligadas a diferentes domínios de conhecimento [Ambrósio 2011]. Alunos ingressantes em cursos como o da computação apresentam comportamentos distintos nas disciplinas introdutórias relacionadas à programação. Enquanto alguns conseguem aprender a programar rapidamente, outros encontram enormes dificuldades [Ambrósio 2011].

Os alunos entendem os enunciados dos exercícios apresentados a eles, no entanto encontram dificuldades em definir uma sequência de comandos que leve ao resultado desejado [Ambrósio 2011]. Durante o desenvolvimento de um algoritmo, muitas informações são produzidas e requeridas, e, muitas vezes, é essencial estabelecer conexões entre recursos de informação a fim de se obter o conjunto necessário de informações para apoiar a realização de alguma atividade [Souza et al. 2010], e, para desempenhar esse papel, pode-se usar as ontologias.

Como um meio de representação, as ontologias são muito utilizadas para representar um contexto específico [Guizzardi 2000]. Além disso, elas possibilitam a

comunicação entre pessoas acerca de determinado conhecimento, pois permitem raciocínio e entendimento sobre um domínio, cuja relação auxilia na obtenção de consenso, principalmente sobre os termos técnicos [Guizzardi 2000].

São diversas as áreas de aplicação das ontologias, tais como: Recuperação de Informações, Gestão do Conhecimento, Educação, Processamento de Linguagem Natural, Mineração de Dados, entre outras. Entre as áreas de aplicação, a *Web* Semântica tem obtido grandes avanços, tornando-se [Berners-Lee 2001] um padrão de vocabulário comum para a troca de dados, o conhecimento reutilizável e a facilidade na comunicação de sistemas heterogêneos [Liu e Özsu 2009].

Sendo assim, este trabalho tem como objetivo elaborar uma ontologia que será modelada a partir de conteúdos da disciplina de algoritmos com a linguagem de programação C, com a finalidade de mostrar informações por meio de consultas e representar o contexto do desenvolvimento desses algoritmos.

1.1. Justificativa

A ontologia é um método de representação do conhecimento muito relevante no campo da educação [Pierrakeas 2012]. As ontologias são consideradas um meio para facilitar a reutilização e o compartilhamento de conhecimento em aplicativos e grupos de pessoas. De acordo com a literatura, elas estão sendo utilizadas para representar domínios de conhecimentos, estratégias de ensino, perfis de estudantes, sistemas de recomendação, entre outros [Pierrakeas 2012].

A ontologia pode ser um modelo interessante para a representação do conhecimento em algoritmos, pois percebe-se a falta de domínio da linguagem de programação por parte dos novos alunos [Ambrósio 2011]. Nesse sentido, disponibilizar informações de estruturas computacionais poderá auxiliar na construção do código.

Sendo assim, este estudo visa implementar uma ontologia, a qual poderá ser utilizada como uma base de conhecimento por ferramentas que entreguem elementos e informações a serem utilizadas na elaboração de algoritmos.

1.2. Objetivo geral

O objetivo geral deste trabalho é modelar uma ontologia para a construção de algoritmos na linguagem de programação C com o uso da ferramenta *Protégé*.

1.3. Objetivos específicos

- Modelar a ontologia para a categorização de informações a partir da disciplina de algoritmos e programação com a ferramenta *Protégé*;
- Utilizar a linguagem SQWRL para realizar consultas na ontologia.

1.4. Estrutura do trabalho

Na seção 2, apresentam-se o referencial teórico e as tecnologias para a realização deste trabalho. Na seção 3, apresentam-se os trabalhos relacionados similares a este estudo. Na seção 4, aborda-se a metodologia utilizada para o desenvolvimento da ontologia e do sistema, e, na seção 5, é apresentada a modelagem da ontologia.

Já na seção 6, apresentam-se as consultas na ontologia e, por fim, na seção 7, as conclusões, as quais tratam de pontos importantes citados neste trabalho e sobre os trabalhos futuros.

2. Referencial teórico

Nesta seção, serão apresentados e discutidos os seguintes temas: *Web* Semântica, ontologia, *Ontology Web Language* (OWL), *Semantic Querry-Enhanced Web Rule Language* (SQWRL), algoritmos e metodologia *Ontology Development 101*.

2.1. *Web* Semântica

A *World Wide Web* foi criada por Tim Barner-Lee, entre 1989 e 1991, baseada nos trabalhos sobre hipertexto realizados por Bush e Ted Nelson. A *Web*, conforme Cunha (2002), surgiu com a visão de que ela seria um espaço em que a informação poderia adquirir um significado bem definido, de forma que facilitasse a cooperação e a comunicação entre as pessoas e os agentes computacionais.

A *Web* Sintática, conforme Breitman (2004), é apenas a apresentação de informações, enquanto o processo de avaliar, classificar e selecionar fica a cargo dos seres humanos. Nesse sentido, os mecanismos de busca são extremamente ricos em quantidade de sites indexados em suas bases de dados, embora sequer os melhores deles consigam abranger a totalidade de conteúdo disponível na *Web* [Pickler 2007]. Apesar da quantidade de informações recuperadas pelos mecanismos, apenas a parte da *Web* é pesquisada, enquanto uma parte considerável do conteúdo fica inacessível [Pickler 2007].

Dessa maneira, a *Web* Semântica visa, justamente, melhorar a satisfação do usuário no momento da busca, retornando-lhe as informações adequadas às suas necessidades. Ao contrário da *Web* sintática, a *Web* Semântica busca capturar o significado das páginas, criando um ambiente em que os computadores possam processar e relacionar conteúdos provenientes de várias fontes. Para isso, é necessário adicionar semânticas aos documentos disponíveis na *Web* [Breitman 2004].

Em outras palavras, na *Web* Semântica, as informações publicadas na rede são preparadas para serem compreendidas tanto por humanos quanto por máquinas, o que resultaria em uma *web* mais eficiente e autônoma na busca e na associação de informações [Lammel e Mielniczuk 2012].

Para a *Web* Semântica tornar-se possível, os computadores necessitam ter acesso a coleções estruturadas de informações e de conjuntos de regras de inferência que ajudem no processo de dedução automática e na representação de conhecimento [Pickler 2007]. Tais regras são especificadas por meio de ontologias que permitem representar a semântica dos dados [Pickler 2007].

2.2. Ontologia

A ontologia teve sua origem na Grécia e significa o estudo da existência de entidades tanto abstratas como concretas que formam o mundo [Turcatel 2014]. Atualmente, na Inteligência Artificial (IA), utiliza-se a ontologia para representar um contexto com grande número de informações [Noy e Mcguinness 2000].

Uma ontologia para a Inteligência Artifical é uma descrição explícita e formal das classes (ou conceitos) em um domínio. Ela também descreve as características, os atributos (ou funções ou propriedades) e as relações entre os membros da classe Noy e Mcguinness 2000]. Uma ontologia, juntamente com um conjunto das instâncias das classes, constitui uma base de conhecimento [Noy e Mcguinness 2000].

Conforme Studer, Benjamins e Fensel (1998) cita, pode-se classificar a ontologia em quatro tipos: ontologias de domínio, ontologias genéricas, ontologias de aplicação e ontologias de representação.

As ontologias de domínio são utilizadas para representar um domínio específico, descrevendo situações reais sobre ele. Já as ontologias genéricas são similares às de domínio, porém suas descrições são mais genéricas, já que estas são independentes de um domínio particular. Por sua vez, as ontologias de aplicação são modeladas para conceitos de uma determinada aplicação. Por último, as ontologias de representação não pertencem a domínio algum, em que apenas se fornecem inferências que serão usadas por outras ontologias, como a de domínio, por exemplo [Studer, Benjamins e Fensel 1998].

2.3. *Ontology Web Language* (OWL)

Para modelar a ontologia, é necessário descrevê-la, a fim de que o computador identifique a linguagem e compreenda as relações. Com isso, a ferramenta *Protégé* utiliza, em seu núcleo, uma linguagem que descreve ontologias [Noy e Mcguinness 2000].

A *Ontology Web Language* (OWL) é uma linguagem para descrever e criar instâncias de ontologias, ou seja, um objeto cujo comportamento e estado são definidos pela ontologia [Pandey, Dwivedi, Verma 2013]. Ela pode incluir descrições de classes, propriedades, entre outras características. Na OWL, está especificada a semântica de maneira formal e como deve-se proceder com os fatos que não estão presentes na ontologia. Atualmente, existem três tipos de sublinguagens da OWL; são elas: OWL *Lite*, OWL *Description Logics* e OWL *Full* [He e An 2011].

A OWL *Lite* é dedicada a usuários que necessitam, de maneira simples, uma classificação e características das relações. Ela só permite valores de cardinalidade de 0 ou 1 e, por ser mais simples, fornece caminhos rápidos entre os dados na ontologia [He e An 2011].

Já a OWL *Description Logics* (OWL DL) foi projetada para usuários que buscam a máxima expressividade, sem perder a completude computacional, em que todas as vinculações são garantidas para serem computadas. Além disso, ela oferece decidibilidade, oferecendo métodos eficientes em problemas de decisão e sendo utilizada para apoiar segmentos em que existam sistemas de raciocínio. Esses sistemas são capazes de simular a habilidade humana de adaptação a experiências anteriores [He e An 2011].

Por fim, a OWL *Full* é destinada a usuários que procuram por máxima expressividade e liberdade sintática, sem garantias computacionais do *Resource Description Framework* (RDF), uma linguagem de representação informal na internet. Ela permite que uma ontologia aumente o significado do vocabulário pré-definido (RDF ou OWL) [He e An 2011].

2.4. SQWRL

Para verificar as informações na linguagem OWL, utilizam-se consultas. Existem diversos tipos de consultas, como a linguagem *XPath (XML Path Language)*, SPARQL (*SPARQL Protocol and RDF Query Language*), SWRL (*Semantic Web Rule Language)*, entre outras.

A linguagem *Semantic Query-Enhanced Web Rule Language* (SQWRL) [O'Connor e Das 2008] permite a criação de consultas a ontologias de forma análoga ao *Structured Query Language* (SQL). Diferentemente de outras consultas em ontologias, as consultas via SQWRL apenas retornam informações da ontologia que satisfazem determinadas restrições e que podem ser manipuladas, posteriormente, utilizando uma linguagem de programação qualquer, como a Java [Gassen et al. 2008].

O SQWRL utiliza uma biblioteca com os métodos do *Semantic Web Rule Language* (SWRL), podendo-se utilizar essas regras para fazer buscas em OWL [Gassen et al. 2008]. Por esse motivo, o SQWRL torna-se uma poderosa linguagem para buscas em OWL, podendo aproveitar ao máximo a expressividade semântica representada em OWL [Gassen et al. 2008], sendo a mais indicada para consultas sobre ontologias expressas nessa linguagem [Gassen et al. 2008].

A sintaxe dos comandos SQWRL é representada da seguinte forma:

Classe (?c) ^ propriedade(?c , ?p) -> sqwrl:select(?c, ?p)

O exemplo acima mostra que os dados antecedidos por "?" são passados ao motor de busca, e o comando "^" significa que será utilizado outro campo de busca. Ao procurar dados, por exemplo, no elemento "propriedade", utilizam-se dois valores. Esses valores são definidos como "*domain*" e "*range*" (domínio e alcance), ou seja, a partir de uma informação, acha-se a outra. Ao declarar qual será a busca, utiliza-se o comando "->" e o comando de busca "*sqwrl:select*", no qual deverão ser selecionados os valores que se desejam retornar da ontologia [Gassen et al. 2008].

2.5. Algoritmos

Um algoritmo pode ser definido como uma sequência finita de passos (instruções) para resolver um determinado problema [Koliver et al. 2009]. São exemplos de algoritmos instruções de montagem, receitas, manuais de uso etc. Um algoritmo não é a solução do problema, pois, se assim fosse, cada problema teria um único algoritmo; um algoritmo é um caminho para a solução de um problema. Em geral, existem muitos (senão infinitos) caminhos que levam a uma solução satisfatória [Tonet e Koliver 2000].

Na computação, um algoritmo é utilizado para fazer com que o computador execute tarefas e, para isso, é necessário que ele execute um programa. Este é um conjunto de instruções que indicam ao computador, passo a passo, o que ele precisa fazer. Logo, um programa é um algoritmo computacional descrito em uma linguagem de programação [Tonet e Koliver 2000].

Existem diversas linguagens de programação, como, por exemplo, a linguagem C, a qual, inicialmente, era utilizada para programação de sistemas. Em virtude de sua portabilidade e eficiência, a linguagem C ganhou grande popularidade [Schildt 1997].

2.6. *Ontology Development 101*

A metodologia utilizada para a construção do modelo ontológico está dividida em sete passos, conforme a *Ontology Development 101*, e foi realizada com o *software Protégé*, o qual possui, no núcleo do *software*, características que reúnem conceitos da OWL DL e da OWL Lite [Noy e Mcguinness 2000]. O objetivo dessa metodologia é a criação de novas ontologias. Algumas características da criação das novas ontologias são: compartilhar entendimento comum da estrutura da informação entre pessoas e agentes de *software;* permitir a reutilização do conhecimento de domínio; e analisar o conhecimento do domínio [Noy e Mcguinness 2000].

Os passos da metodologia são:

1. Determinação do domínio e do escopo da ontologia: de acordo com a metodologia, é necessário que se respondam algumas questões para definir qual o escopo da ontologia. São elas: que domínio a ontologia irá cobrir? Por que usar ontologia? Para quais tipos de questões a informação presente na ontologia deve prover respostas? Quem irá utilizar e manter a ontologia? Além dessas perguntas, são necessárias questões mais específicas, como: qual tipo de variável devo usar?

2. Reutilização das ontologias: uma das vantagens de reutilizar ontologias é a possibilidade de tornar o modelo compartilhável, já que modelos que utilizam ontologias reutilizadas tornam-se mais facilmente integrados. É sempre possível utilizar o que outras pessoas já fizeram, já que a proposta deste trabalho é justamente esta (esse passo não é obrigatório).

3. Enumeração dos termos importantes na ontologia: os termos, geralmente, aparecem em sentenças ou devem ser explicados ao usuário. A metodologia utiliza algumas questões para auxiliar na criação desses termos, tais como: sobre quais termos quer-se falar? Quais propriedades têm esses termos? O que falar sobre esses termos? O que geralmente ocorre é que se define um conjunto pequeno de classes, definido como um conjunto de objetos com características similares, e, logo em seguida, definem-se suas propriedades. Após definir suas propriedades, criam-se novas subclasses e, depois, definem-se, novamente, suas propriedades, e assim por adiante.

4. Definição de classes e hierarquia de classes: existem várias abordagens para desenvolver uma hierarquia de classes, como: *Top-down* – primeiro, são definidos os conceitos mais genéricos do domínio da etapa 1 e, posteriormente, são definidos os conceitos mais específicos; *Bottom-up* – primeiro, são definidos os conceitos específicos do domínio, também estabelecidos na etapa 1, e, na sequência, são estabelecidos os mais genéricos; *Hibrida* – é uma combinação das duas hierarquias anteriores, em que são definidos os conceitos mais importantes do domínio e, após a definição, eles serão especializados ou generalizados conforme o caso.

Para iniciar as definições, utiliza-se a lista de termos estipulada na etapa 3. Então, serão identificados os termos independentes em vez dos termos que descrevem características desses objetos. Tais termos serão definidos como classes, superclasses e instâncias, cujos comportamentos e estados são definidos pela classe.

5. Definição das propriedades das classes: essas definições somente não são suficientes para responder as perguntas realizadas na etapa 1, e, por isso, também deverá ser definida a estrutura interna dos conceitos. Após a seleção de termos que se tornarão classes, executada no passo 3, selecionam-se os termos restantes para a descrição de

propriedades dessas classes. Exemplos de propriedade: ser de natureza intrínseca ou extrínseca ao conceito a que são relativas; descrever partes físicas ou abstratas do conceito; e representar relacionamentos dos indivíduos pertencentes a uma determinada classe com indivíduos de outra. A propriedade atribuída à classe deve estar relacionada à classe mais genérica, pois todas as classes derivadas desta herdarão tal propriedade.

6. Definição das características/restrições das propriedades: a propriedade de uma classe poderá ter várias características, como cardinalidade, tipo do valor e domínio e alcance da propriedade. Nesse passo, devem-se atribuir as características intrínsecas e extrínsecas da ontologia.

7. Criação das instâncias: esse é o último passo para a criação de uma ontologia. Nele, são criadas instâncias individuais de uma classe. Para isso, deve-se escolher uma classe, criar a instância individual da classe selecionada e preencher os valores das propriedades.

3. Trabalhos relacionados

Foram identificados vários trabalhos relacionados, dos quais se citam somente três, considerados relevantes por utilizarem a ontologia de forma similar à utilizada neste estudo.

O estudo realizado por Lee, Ye e Wang (2005) aborda a modelagem de uma ontologia voltada à aprendizagem na linguagem Java. No trabalho do autor, utilizam-se informações baseadas em uma matriz curricular para ambientes de autoaprendizagem para cursos introdutórios de programação. O objetivo do autor com a ontologia é proporcionar a autoaprendizagem.

Já o trabalho de Pierrakeas (2012) desenvolve uma ontologia que se baseia no aprendizado de linguagens de programação. Utilizando a metodologia *Ontology Development 101*, o autor descreve duas linguagens como exemplos (linguagem C e Java*)*. Com o objetivo de ensinar a programação a possíveis alunos e facilitar a representação das linguagens pelos tutores, o autor descreve as funcionalidades e as características de cada uma das linguagens por meio de descrições e informações.

Por sua vez, o trabalho de Abuhassan (2012) realiza a modelagem de uma ontologia para descrever linguagens de programação orientadas a objeto. O objetivo do autor é auxiliar pesquisadores no desenvolvimento e no estudo das linguagens já existentes, além de centralizar informações sobre essas linguagens. Para isso, ele descreve as linguagens de programação com características específicas de cada uma delas, categorizando-as com o objetivo de cada uma.

Este estudo difere dos trabalhos dos autores, pois, além de representar a linguagem de programação C, também visa representar exercícios dados a alunos. Ele tem como objetivo representar a utilização da linguagem frente a exercícios, visto que, conforme Pierrakeas (2012), a linguagem C é uma das primeiras a ser apresentadas para novos estudantes do curso de computação. Por meio de consultas, é possível verificar, em exercícios propostos, a estrutura e a descrição de elementos da linguagem C para auxiliar o aluno na compreensão do conteúdo.

4. Metodologia

A modelagem da ontologia foi realizada com a metodologia *Ontology Development 101*, utilizando a ferramenta *Protégé* na versão 5.0.0. Essa ferramenta utiliza, em seu núcleo, a linguagem OWL e permite, de forma estruturada, manipular as entidades da ontologia e realizar consultas na mesma, usando o *Protégé*. Além disso, na metodologia, os autores recomendam e utilizam essa ferramenta.

O conteúdo da ontologia é abordado a partir das disciplinas de Algoritmos I e II, no curso de graduação da área da Informática do Centro Universitário Franciscano. A ontologia baseia-se na linguagem C, pois, nessas disciplinas, os algoritmos são trabalhados e implementados nessa linguagem.

Para as consultas na ontologia, foi utilizada a linguagem SQWRL. Ela foi baseada nas relações dos indivíduos que representam as questões utilizadas na disciplina, consultando as descrições, os valores e a estrutura de cada indivíduo, a fim de mostrar cada elemento.

5. Modelagem da ontologia

No primeiro passo da metodologia *Ontology Development 101*, deve-se responder algumas perguntas para definir o domínio da ontologia. As perguntas e respostas podem ser visualizadas na tabela 1.

Tabela 1. Perguntas e respostas elaboradas para a modelagem da ontologia

Perguntas	Respostas
Que domínio a ontologia irá cobrir?	As disciplinas de Algoritmos I e II do Centro Universitário Franciscano.
Por que usar ontologia?	Para a identificação de padrões na construção de algoritmos e de exercícios propostos, cujo objetivo é mostrar as estruturas envolvidas e as soluções dos exercícios.
Para que tipos de perguntas a informação na ontologia deve fornecer respostas?	Para dúvidas na construção e desenvolvimento dos códigos.
Quem vai usar e manter a ontologia?	Professores da disciplina selecionada e o autor deste projeto.

Após responder as questões acima, a metodologia recomenda criar algumas questões para complementar o escopo da ontologia. Essas questões foram respondidas e ajudaram a formar o domínio da ontologia. Algumas questões desenvolvidas são descritas abaixo:

- *Int* armazena números ou caracteres?
- Qual a melhor declaração para questões com palavras?
- Que tipo de variável é ideal para questões de cálculo?
- Quais as características de uma variável do tipo *float*?
- Quais as características de uma variável do tipo c*har*?
- Quais as características de uma variável do tipo *int*?
- Em quais tipos de problemas deve-se usar uma variável do tipo *int*?
- Em quais tipos de problemas deve-se usar uma variável do tipo c*har*?

- Em questões para contar o número de letras na palavra, o que usar?
- Em quais tipos de questões de cálculo deve-se usar o f*or*?
- Quais as características do *if*?
- Quais as características do *else*?
- Para questões com duas possibilidades, qual estrutura deve-se usar?
- Quando se deve usar o *else*?
- Quais as características do *if else*?
- Quais as características do *printf*?
- Quais as características do *scanf*?
- Para questões de cálculo, qual estrutura usar?
- Quais as características do laço de repetição?
- Para contar letras ou números, o que usar?

O segundo passo da metodologia é a reutilização de ontologias. Existem diversas bibliotecas de ontologias reutilizáveis na *Web* e algumas comerciais. Para este estudo, não será reutilizada ontologia alguma, pois a proposta deste trabalho envolve o desenvolvimento de uma ontologia desde o seu início.

No terceiro passo, foram selecionados termos (Tabela 2) considerados importantes sobre os questionamentos realizados no passo um, e outros foram retirados de bibliografias, os quais são adicionados para ajudar a completar o domínio que a ontologia representará [Schildt 1997] [Koliver et al. 2009].

Tabela 2. Termos selecionados para serem utilizados como superclasses, classes e subclasses

Termos		
Abrindo Arquivos	Funções Data e Hora	Operadores Aritméticos
Abrir e Fechar	Funções de Alocação Dinâmica	Operadores de Ponteiros
Algoritmos	Funções de E/S	Operadores Relacionais e Lógicos
Alocação Dinâmica	Funções Gráficas	Outros Tipos de Dados
Argumentos de Funções	Funções Matemáticas	Parâmetros Formais
Argumentos para main	Funções Miscelâneas	Parênteses e Colchetes
Arquivos	Funções String e Caracteres	Ponteiro para Matriz
Arquivos Separados	Funções tipos Void	Ponteiros
Atribuições Múltiplas	Globais	Ponteiros de Arquivos
Bibliotecas	Ímpar ou Par	Ponteiros e Matrizes
Bibliotecas e Arquivos	Imprimir Palavras	Print
Blocos de Comandos	Incremento e Decremento	Protótipos
Comandos	Indexando Ponteiro	Questões Aritméticas
Comandos de Desvio	Inserindo Inteiros Sem Sinal	Questões de Arquivos
Comandos de Iteração	Inserindo Números	Questões de Estrutura Repetição
Comandos de Seleção	Lendo Arquivo	Questões de Vetores e Matrizes
Console	Lendo Caracteres	Questões Sintáticas
Constantes	Lendo Strings	Questões de Estrutura If Else
Constantes Caracteres de Barra Invertida	Locais	Recursão
Constantes Hexadecimais e	Local de Declaração	Retornando Ponteiros

Octais		
Constantes String	Matrizes de Ponteiros	Return
Conversão de Tipos em Atribuições	Matrizes e String	Scan
Escrevendo Arquivo	Matrizes Multidimensionais	Sequências
Escrevendo Caracteres	Matrizes unidimensionais	String
Escrevendo Endereço	Matrizes Unidimensionais para funções	Tamanho da Palavra
Escrevendo Números	Operações Matemáticas	Tipos
Escrever e Ler	Operador Bit a Bit	Tipos de dados
Exercícios	Operador de Atribuição	Variáveis
Expressões	Operador Ponto e Seta	Variáveis Ponteiros
Expressões com ponteiros	Operador sizeof	Verdadeiro e Falso
Fechando Arquivos	Operador Vírgula	Vogal ou Consoante
Funções	Operadores	

Em seguida, no passo quatro, é realizada a elaboração da hierarquia das classes. Para isso, primeiramente, é necessário definir as superclasses, as classes e as subclasses da ontologia. Essa definição será baseada na sua generalidade. As superclasses escolhidas foram "Algoritmos" e "Exercícios", pois são mais genéricas e, sendo assim, representam uma gama maior de informações.

Já as classes escolhidas foram selecionadas por não serem tão genéricas quanto às superclasses e menos específicas que as subclasses (Tabela 3).

Tabela 3. Termos selecionados para serem utilizados como classes

Classes		
Arquivos	Matrizes e String	Questões de Estrutura Repetição
Comandos	Ponteiros	Questões de Vetores e Matrizes
Console	Questões Aritméticas	Questões Sintáticas
Expressões	Questões de Arquivos	Questões de Estrutura If Else
Funções		

Os demais termos do passo três foram utilizados para a escolha das subclasses. Esses termos são mais específicos e relacionados às classes.

Como etapa final do passo quatro, foi elaborada a hierarquia das classes (Figura 1). Para tal, utilizou-se o modelo *Top-down*, pois ele apresenta uma visualização partindo dos termos mais genéricos para os mais específicos.

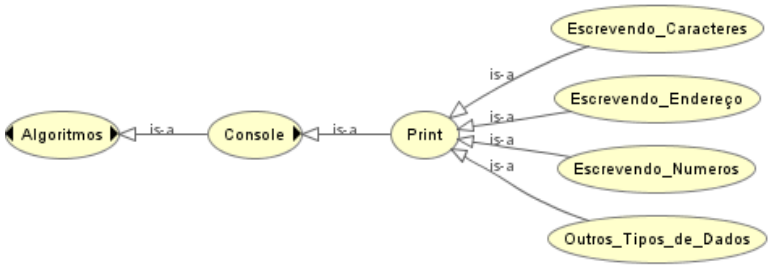

Figura 1. Hierarquia *Top-Down* dos termos escolhidos como classes e subclasses

Conforme mostra a figura 1, as subclasses "Escrevendo_Endereço" e "Escrevendo_Caracteres", por exemplo, são classes mais específicas das classes "*Print*",

“Console” e “Algoritmos”. Essas subclasses são tipos de “*print”* que a linguagem C possui. Nessa sequência, verifica-se que a classe “Algoritmos” é a mais genérica, e as subclasses “Escrevendo_Endereço” e “Escrevendo_Caracteres” são mais específicas.

No quinto passo, foram definidas as propriedades das classes. Nessa etapa, definem-se as propriedades intrínsecas e extrínsecas da ontologia. As propriedades intrínsecas foram definidas para realizar o relacionamento entres as classes e as instâncias. Na tabela 4, estão demonstradas todas as propriedades intrínsecas da ontologia. É possível visualizar que a propriedade “Tem Algoritmo”, por exemplo, relaciona a classe “Problemas” com a classe “Algoritmos”. Essa relação é definida pela expressão: “Problemas Tem Algoritmo Algoritmos”. Na ontologia, há, também, propriedades intrínsecas que relacionam indivíduos, cuja relação define que um indivíduo possui outro indivíduo como uma propriedade.

Tabela 4. Propriedades intrínsecas definidas como relacionamento da ontologia

Classe	Relacionamento	Alcance
Exercícios	Tem Algoritmo	Algoritmos
Algoritmos	Manipula Arquivos	Arquivos
Arquivos	Funções de Arquivos	Subclasses de Arquivos
Algoritmos	Manipula Matrizes e Strings	Matrizes e String
Matrizes e String	Propriedades Matrizes e String	Subclasses de Matrizes e String
Algoritmos	Manipula Ponteiros	Ponteiros
Ponteiros	Propriedades de Ponteiros	Subclasses de Ponteiros
Ponteiros e Matrizes	Características das Matrizes de Ponteiros	Matrizes de Ponteiros
Algoritmos	Tem Comandos	Comandos
Comandos	Tipos de Comandos	Subclasses de Comandos
Algoritmos	Tem Funções	Funções
Funções	Propriedades das Funções	Subclasses de Funções
Bibliotecas e Arquivos	Propriedades Bibliotecas e Arquivos	Subclasses de Bibliotecas e Arquivos
Tipos	Tipos de Funções	Subclasses de Tipos
Algoritmos	Usa Console	Console
Console	Manipular Informações	Print, Scan
Print	Propriedades Print	Subclasses de Print
Scan	Propriedades Scan	Subclasses de Scan
Algoritmos	Usa Expressões	Expressões
Expressões	Tipos Expressões	Subclasses de Expressões
Constantes	Propriedades Constantes	Subclasses de Constantes
Operadores	Propriedades Operador	Subclasses de Operadores
Variáveis	Propriedades Variáveis	Subclasses de Variáveis
Local de Declaração	Variáveis Declaração	Subclasses de Local de Declaração
Algoritmos	Tem Problemas	Problemas
Exercícios	Problemas Tipo	Subclasses de Exercícios
Questões	Tipo Aritméticos	Subclasses de

Aritméticas		Questões Aritméticas
Questões de Arquivos	Tipo Arquivo	Subclasses de Questões de Arquivos
Questões Aritméticas	Tipo Sintático	Questões Aritméticas
Indivíduos	Tem Elementos	Indivíduos

Ainda no quinto passo, tem-se a criação de propriedades extrínsecas (Tabela 5). Estas definem os conteúdos que estão fora do contexto da classe, como, por exemplo, descrição, valor, entre outros. Essas propriedades são atribuídas às instâncias da ontologia. Tais propriedades definem atributos complementares.

Tabela 5. Propriedades extrínsecas dos indivíduos da ontologia

Propriedade	Objetivo
Tem Descrição	Atribui a um indivíduo uma descrição.
Tem Estrutura	Atribui a um indivíduo a forma estruturada a qual é usado.
Tem Valor	Atribui a um indivíduo um valor.
Tem Narrativa	Atribui aos indivíduos de questionamentos a narrativa da questão.

No sexto passo, devem-se atribuir as propriedades criadas no passo cinco aos elementos da ontologia (Figura 2). Atribuem-se as propriedades intrínsecas às superclasses com as classes e subclasses ou de indivíduo para indivíduo. Essas propriedades não possuem cardinalidade definida, e seu alcance é de direta relação, ou seja, relaciona a superclasse com a classe, a classe com a subclasse, e assim sucessivamente. As propriedades extrínsecas são atribuídas apenas aos indivíduos e possuem valores do tipo "*String*", pois são descritas determinadas informações a esses indivíduos.

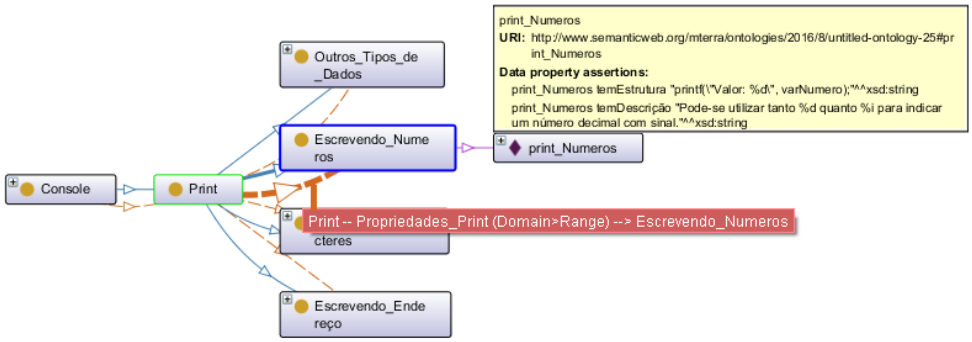

Figura 2. Propriedades atribuídas à classe "Escrevendo_Numeros" e à instância "print_Numeros".

A última etapa da modelagem é a criação das instâncias. As instâncias são parte de uma única classe, as quais definirão um indivíduo da classe. Na Figura 3, é possível visualizar um exemplo de três instâncias atribuídas à classe "Comandos_de_Iteração" da qual esses indivíduos fazem parte. Assim, um indivíduo "*For*", por exemplo, é um comando de iteração.

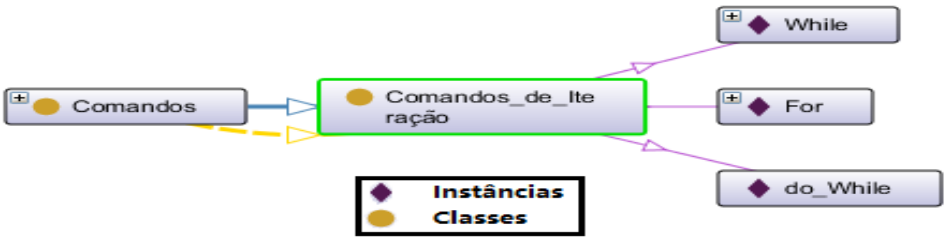

Figura 3. Instâncias atribuídas à classe "Comandos_de_Iteração".

Todas as instâncias elaboradas para a modelagem da ontologia possuem pelo menos uma característica estabelecida pelas propriedades extrínsecas, contendo informações que serão coletadas pela consulta.

Após a definição das superclasses, classes, subclasses, instâncias, características intrínsecas e extrínsecas e seus relacionamentos, tem-se a modelagem da ontologia. Na figura 4, tem-se a visualização de todas as classes relacionadas na ontologia.

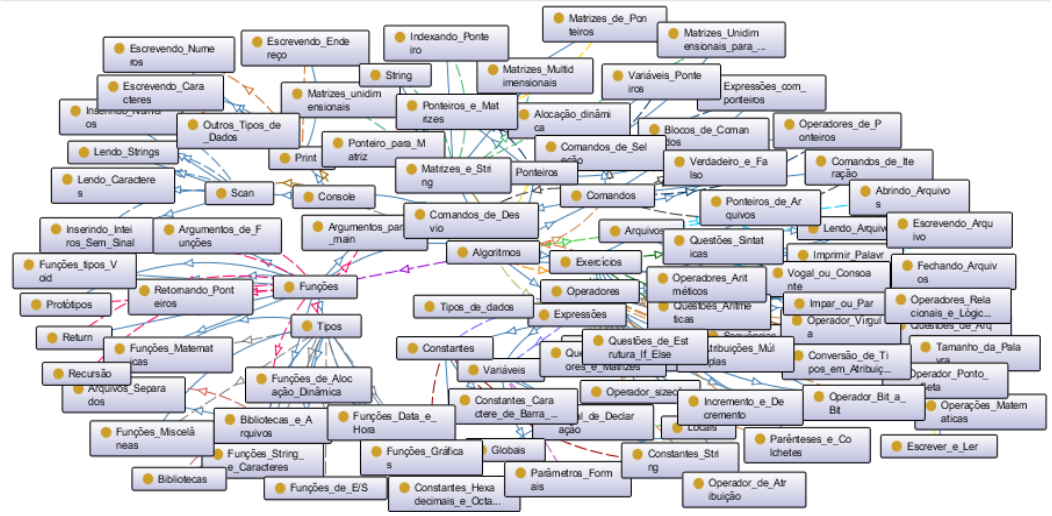

Figura 4. Visão geral da ontologia

6. Consultas

Para verificar as informações atribuídas na modelagem da ontologia e responder as questões criadas no passo um, utilizam-se consultas que buscam, na ontologia, informações que respondem os questionamentos. Essas consultas foram elaboradas utilizando a linguagem SQWRL. Nelas, buscam-se, nas classes, os indivíduos de interesse, mostrando suas propriedades (descrição, valor etc.). Nesse sentido, tem-se como exemplo a consulta abaixo, que responde o questionamento: "qual é a característica do comando "*if*"? ":

Comandos(if) ^ temDescrição(if, ?descrição) ^ temEstrutura(if, ?estrutura) -> sqwrl:select(?descrição, ?estrutura)

Figura 5. Consulta do Comando if

Essa consulta busca o indivíduo "*if*" na classe a que ele pertence. Buscam-se, também, as suas propriedades extrínsecas, por meio da propriedade "temDescrição" e "temEstrutura". Para isso, as variáveis "descrição" e "estrutura" devolvem a

característica do indivíduo selecionado. Como resultado, tem-se o conteúdo do indivíduo:

?descrição
A estrutura if é utilizada para verificar se uma determinada expressão é satisfatória. Caso o seja, ela executa um determinado comando.

Figura 6. Retorno da consulta pela variável "descrição"

?estrutura
A forma geral da sentença if é: ***if(expressão) comando;*** ***else comando;*** *onde comando pode ser um único comando, um bloco de comandos ou nada (no caso de comandos vazios). A cláusula else é opcional.*

Figura 7. Retorno da consulta pela variável "estrutura"

Cada consulta possui um tipo de retorno especificado um a um. Na ontologia, também se descreveram exercícios. Cada exercício descrito na ontologia procurou mostrar as estruturas utilizadas, bem como suas descrições, a fim de mostrar o domínio de cada exercício em relação à linguagem de programação. Neste estudo, foram criados diversos tipos de exercícios, desde questões aritméticas até questões de arquivos. Para responder, por exemplo, o questionamento "qual estrutura utilizar em questões de cálculo? ", tem-se a consulta abaixo:

Questões_Aritméticas(Soma_de_Numeros_Pares) ^ temElementos(Soma_de_Numeros_Pares, ?elemento) ^ temDescrição(?elemento, ?descrição) -> sqwrl:select(?elemento, ?descrição)

Figura 8. Consulta para verificar qual estrutura utilizar em questões de cálculo

Essa consulta busca o exercício "Soma_de_Numeros_Pares". Para isso, busca-se, na propriedade intrínseca "temElementos", todos os indivíduos ligados ao questionamento. Essa propriedade é consultada e retornada pela variável "elemento". Para descrever cada um dos indivíduos, é consultada a propriedade extrínseca "temDescrição" de cada um dos elementos retornados por meio da variável "elemento". Esta propriedade retorna à descrição de cada um dos elementos pela variável "descrição". Ao final, tem-se a listagem de todos os elementos e suas descrições, à qual pertence a resposta do exercício, representada pelo indivíduo "Soma_de_Numeros_Pares" (Apêndice A). Para a ontologia, criou-se trinta consultas para validar as informações, descrições e valores da modelagem (Apêndice B).

7. Conclusões e trabalhos futuros

Este estudo apresentou a modelagem de uma ontologia que representa os conteúdos das disciplinas de algoritmos com a linguagem C frente a exercícios.

Diante das diversas possibilidades existentes para o auxílio no desenvolvimento de algoritmos, optou-se pela modelagem de uma ontologia justamente por ela permitir o compartilhamento de informação e sua reutilização. Além disso, a ontologia serve de base de conhecimento de diversos sistemas, os quais buscam, na ontologia, relações

entre as informações e, a partir disso, trabalham melhor com a informação contida no ambiente.

A respeito da metodologia utilizada no desenvolvimento da ontologia, optou-se pela *Ontology Development* 101 por ela apresentar passos detalhados para a modelagem e uma ferramenta de desenvolvimento.

Com a ontologia desenvolvida neste estudo, permite-se aos estudantes buscar informações com maior facilidade frente aos desafios que lhes são apresentados no desenvolvimento dos algoritmos, pois as informações estão centralizadas e compartilhadas. Com isso, o desenvolvimento pode tornar-se mais dinâmico, permitindo que dúvidas como o que utilizar ou como utilizar determinados elementos da linguagem sejam definidas pela ontologia.

Durante o trabalho, encontraram-se algumas dificuldades, como problemas de desempenho e limitações técnicas com a utilização da ferramenta *Protégé* e com informações a respeito da estrutura das consultas na ontologia. Além disso, este estudo possui algumas limitações, como a necessidade de outro *software* para a utilização da ontologia em um sistema que auxilie no desenvolvimento dos algoritmos.

Como trabalho futuro, pretende-se utilizar a ontologia como uma base de informação para o desenvolvimento de um sistema de recomendação com o *framework* Jena. Esses sistemas buscam informações a partir de um determinado conhecimento para recomendá-los a usuários. Também se pretende aplicar essa ontologia em uma disciplina de algoritmos, a fim de verificar sua utilização junto aos estudantes.

8. Referências

Abuhassan, I. A. O. and AlMashaykhi, A. M. O. (2012) Domain Ontology for Programming Languages. In *Journal of Computations & Modelling*, pages 75-91.

Ambrósio, A. P. L., Almeida, L. S., Macedo, J., Santos, A. e Franco, A. H. (2011) Programação de computadores: compreender as dificuldades de aprendizagem dos alunos. In *Revista Galego-Portuguesa de Psicoloxia e Educación, páginas 185-197.*

Berners-Lee, J. H. T. and Lassila, O. (2001) "The semantic web*",* https://www.scientificamerican.com/article/the-semantic-web/#, junho.

Breitman, K. e Leite, J. C. S. P. (2004) "Ontologias: como e por que criá-las", http://www-di.inf.puc-rio.br/~julio/Slct-pub/JAI.pdf, maio.

Cunha, L. M. S. (2002) "*Web* Semântica: estudo preliminar". Embrapa Informática Agropecuária. https://www.embrapa.br/busca-de-publicacoes/-/publicacao/8670/web-semantica-estudo-preliminar, setembro.

Gassen, J. B., Freitas, L. O., Silveira, M. C. e Librelotto, G. R. (2008) "Uma comparação entre linguagens para consultas sobre ontologias OWL". UNIFRA – Centro Universitário Franciscano, Seminário de Informática – RS.

Guizzardi, G. (2000) "Desenvolvimento para e com reuso: um estudo de caso no domínio de vídeo sob demanda". Dissertação de Mestrado. Centro Tecnológico da Universidade Federal do Espírito Santo.

He, G. and An, L. (2011) "Ontology Language OWL Research Study". BaoDing

University, BaoDing, China.

Koliver, C. et al. (2009) “Introdução à construção de algoritmos: notas de Aula”. Editora Educs, Caxias do Sul, Brasil.

Lammel, I. e Mielniczuk, L. (2012) Aplicação da *Web* Semântica no Jornalismo. In *Estudos em Jornalismo e Mídia*, https://periodicos.ufsc.br/index.php/jornalismo/article/viewFile/1984-6924.2012v9n1p180/22315, outubro.

Lee, M., Ye, D. Y. and Wang, T. I. (2005). “Java Learning Object Ontology”. Laboratory of Intelligent Network Applications, National Chung Kung University. http://ieeexplore.ieee.org/document/1508750/?reload=true, outubro.

Liu, L. and Özsu, M. T. (2009) “Encyclopedia of Database Systems”, v. 1., Editora Springer.

Noy, N. F. and Mcguinness, D. L. (2000) “Ontology Development 101: a guide to creating your first ontology”. Stanford University, http://protege.stanford.edu/publications/ontology_development/ontology101.pdf, maio.

Pandey, R., Dwivedi, D. S. and Verma, P. (2013) “UnivPeopleProgram Ontology: a OWL based Structural definition for Semantic Web”. Uttar Pradesh, http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6558198, outubro.

Pickler, M. E. V. (2007) “*Web* Semântica: ontologias como ferramentas de representação do conhecimento”. Universidade Estadual de Londrina. http://portaldeperiodicos.eci.ufmg.br/index.php/pci/article/view/251, setembro.

Pierrakeas, C., Solomour, G. and Kameas, A. (2012) An Ontology-based Approch in Learning Programming Languages. In *16th Panhellenic Conference on Informatics,* http://ieeexplore.ieee.org/document/6377424/?reload=true, October.

O’ Connor, M. J. and Das, A. K. (2008) “SQWRL: a query language for OWL”. Stanford Center for Biomedical Informatics Research, http://ceur-ws.org/Vol-529/owled2009_submission_42.pdf, setembro.

Schildt, H. (1997) “C Completo e Total”. Editora E.R.J. Informática Ltda, 3ª Edição.

Souza, F. P., Zampirolli, F. A., Pimentel, E. P. e Marietto, M. G. B. (2010) “Estudo de Ontologias em Engenharia de *Software*”, http://ic.ufabc.edu.br/II_SIC_UFABC/resumos/paper_5_291.pdf, setembro.

Studer, R., Benjamins, V. R. and Fensel, D. (1998) “Knowledged Engineering: principles and methods”, http://www.sciencedirect.com/science/article/pii/S0169023X97000566, maio.

Tonet, B. e Koliver, C. (2000) “Núcleo de Apoio Aprendizagem de Programação”. Universidade de Caxias do Sul, https://www.ucs.br/portais/cent/nucleos/568/, outubro.

Turcatel, I. O. (2014) “Ontologias para a aescoberta de recursos na ciência: análise de VIVO-ISF”. Trabalho de conclusão de curso, Universidade Federal Do Rio Grande Do Sul, http://www.lume.ufrgs.br/handle/10183/112165, setembro.

Apêndice A. Imagem do resultado da consulta “Soma_de_números_pares”

elemento	descrição
scanf_-_Inserindo_Numeros	Utilizado para ler qualquer tipos de números.
operador_atribuição	Este operador, é responsável por atribuir um valor a uma váriavel.
print_Numeros	Pode-se utilizar tanto %d quanto %i para indicar um número decimal com sinal.
For	O laço For é utilizado como uma maneira de, a partir de uma condição, realizar comandos que, se não satisfação a condição, prossig...
Inteiro	Armazena valores inteiros positivos e negativos.
variavel_local	Variavel Local é declarada dentro da função principal do código (main) ou de métodos.
ifs_aninhados	Um if aninhado é um comando if que é o objeto de outro if ou else.
Operador_Relacional_e_Logico	Operador relacional, refere-se as relações que os valores podem ter uns com os outros. Operador lógico, refere-se as maneiras como ...
incremento_e_decremento	Operador " -- " para decrementar ou "++" para incrementar.
Operador_Aritmético	Operadores aritméticos são utilizados para funções matematicas.

Apêndice B. Consultas realizadas na ontologia

Name	Query	Comment
Abrir e Fechar arquivo.	temElementos(Abrir_e_Fechar, ?elemento) ^ temDescrição(?elemento, ?descrição) ^ Questões_de_Arquivos(Abrir_e_Fechar) -> sqwrl:select(?elemento, ?descrição)	Estrutura de questões que envolve abrir e fechar arquiv...
Acha problema.	Exercícios(?arq) ^ temElementos(?arq, For) ^ temNarrativa(?arq, ?Narrativa) -> sqwrl:select(?arq, ?Narrativa)	Descrições das questões.
Caracter	temDescrição(Caracter, ?descrição) ^ temEstrutura(Caracter, ?estrutura) ^ temValor(Caracter, ?valor) ^ Tipos_de_dados(Caracter) -> sqwrl:select(Caracter, ?descrição, ?estrutura, ?...	Descrição do tipo de váriavel caracter (char).
Classes.	tbox:cd(?c) -> sqwrl:select(?c)	Mostra todas as classes da ontologia.
Comandos	Comandos(?com) ^ temDescrição(?com, ?descrição) ^ temEstrutura(?com, ?estrutura) -> sqwrl:select(?com, ?descrição, ?estrutura)	Mostra os comandos do algoritmos.
Cálculo da Potência.	Questões_Aritméticas(Calcular_Potencia) ^ temDescrição(?elemento, ?descrição) ^ temElementos(Calcular_Potencia, ?elemento) -> sqwrl:select(?elemento, ?descrição)	Questão sobre o cálcula a potência de um número.
Cálculo do Seno	temDescrição(?elemento, ?descrição) ^ temElementos(Calcular_Seno, ?elemento) ^ Questões_Aritméticas(Calcular_Seno) -> sqwrl:select(?elemento, ?descrição)	Questão sobre o Seno de um ângulo.
Double	temValor(Precisão_Dupla, ?valor) ^ Tipos_de_dados(Precisão_Dupla) ^ temDescrição(Precisão_Dupla, ?descrição) ^ temEstrutura(Precisão_Dupla, ?estrutura) -> sqwrl:select(Preci...	Descrição do tipo de váriavel de precisão dupla (doubl...
Duas possibilidades.	Exercícios(Verifica_Se_CPF_ou_CNPJ) ^ temElementos(Verifica_Se_CPF_ou_CNPJ, ?elementos) ^ temDescrição(?elementos, ?descrição) -> sqwrl:select(?elementos, ?descrição)	Estrutura com exercício com duas possibilidades.
Else if.	Comandos(ifs_aninhados) ^ temEstrutura(ifs_aninhados, ?estrutura) ^ temDescrição(ifs_aninhados, ?descrição) -> sqwrl:select(?descrição, ?estrutura)	Estrutura do Else if.
Estruturas de Algoritmos.	Algoritmos(?alg) ^ temDescrição(?alg, ?descrição) ^ temEstrutura(?alg, ?estrutura) -> sqwrl:select(?alg, ?descrição, ?estrutura)	Mostra todos os elementos modelados em algoritmos.
Exercícios For.	temNarrativa(?e, ?narrativa) ^ Exercícios(?e) ^ temElementos(?e, For) -> sqwrl:select(?e, ?narrativa)	Exercícios que utilizam o For.
Expressões.	Expressões(?exp) ^ temDescrição(?exp, ?descrição) ^ temEstrutura(?exp, ?estrutura) -> sqwrl:select(?exp, ?descrição, ?estrutura)	Mostra as expressões do algoritmo.
Fibonacci	temElementos(Fibonacci, ?elemento) ^ temDescrição(?elemento, ?descrição) ^ Questões_Aritméticas(Fibonacci) -> sqwrl:select(?elemento, ?descrição)	Mostra estrutura dos questionamentos de Fibonacci.
Funções.	Tipos(?função) ^ temDescrição(?função, ?descrição) ^ temEstrutura(?função, ?estrutura) -> sqwrl:select(?função, ?descrição, ?estrutura)	Mostra as funções dos algoritmos.
IF	temDescrição(if, ?descrição) ^ Comandos(if) ^ temEstrutura(if, ?estrutura) -> sqwrl:select(?descrição, ?estrutura)	Estrutura do if.
Impar ou Par.	temDescrição(?elemento, ?descrição) ^ Questões_Aritméticas(Impar_ou_Par) ^ temElementos(Impar_ou_Par, ?elemento) -> sqwrl:select(?elemento, ?descrição)	Consulta a qual mostra os elementos de um questiona...
Indivíduos.	owl:Thing(?i) -> sqwrl:select(?i)	Mostra todos os indivíduos.
Inteiro	temValor(Inteiro, ?valor) ^ Tipos_de_dados(Inteiro) ^ temDescrição(Inteiro, ?descrição) ^ temEstrutura(Inteiro, ?estrutura) -> sqwrl:select(Inteiro, ?descrição, ?estrutura, ?valor)	Descrição do tipo de váriavel inteiro (int).
Laços de repetição	temEstrutura(?comandos, ?estrutura) ^ Comandos_de_Iteração(?comandos) ^ temDescrição(?comandos, ?descrição) -> sqwrl:select(?comandos, ?descrição, ?estrutura)	Mostra a estrutura dos laços de repetição.
Número de Letras - Estrutura.	temElementos(Quantidade_de_Letras, ?elementos) ^ Questões_Sintaticas(Quantidade_de_Letras) ^ temDescrição(?elementos, ?descrição) -> sqwrl:select(?elementos, ?descrição)	Descreve a estrutura das questões com números de let...
Números Pares	temElementos(Soma_de_Numeros_Pares, ?elemento) ^ temDescrição(?elemento, ?descrição) ^ Questões_Aritméticas(Soma_de_Numeros_Pares) -> sqwrl:select(?elemento, ?desc...	Descreve a estrutura das questões com números pares.
Ponto Flutuante	temEstrutura(Ponto_Flutuante, ?estrutura) ^ Tipos_de_dados(Ponto_Flutuante) ^ temValor(Ponto_Flutuante, ?valor) ^ temDescrição(Ponto_Flutuante, ?descrição) -> sqwrl:select(...	Descrição do tipo de váriavel de ponto flutuante (float).
Primos	temElementos(Fibonacci, ?elemento) ^ temDescrição(?elemento, ?descrição) ^ Questões_Aritméticas(Primos) -> sqwrl:select(?elemento, ?descrição)	Descreve a estrutura das questões com números primos.
Print	Print(?p) ^ temEstrutura(?p, ?estrutura) ^ temDescrição(?p, ?descrição) -> sqwrl:select(?p, ?descrição, ?estrutura)	Descreve a estrutura dos prints.
Quando usar IF e Else.	temNarrativa(?e, ?narrativa) ^ Exercícios(?e) ^ temElementos(?e, ifs_aninhados) -> sqwrl:select(?e, ?narrativa)	Mostra as questões que utiliza-se if e else.
Questões com Palavras.	temElementos(Quantidade_de_Letras, ?elementos) ^ Questões_Sintaticas(Quantidade_de_Letras) ^ temDescrição(?elementos, ?descrição) -> sqwrl:select(?elementos, ?descrição)	Mostra os elementos das questões com palavras.
Questões de cálculo.	temElementos(?op, ?elementos) ^ Operações_Matematicas(?op) ^ Tipos_de_dados(?elementos) -> sqwrl:select(?elementos)	Tipos de dados com questões de cálculo.
Scan	Scan(?s) ^ temDescrição(?s, ?descrição) ^ temEstrutura(?s, ?estrutura) -> sqwrl:select(?s, ?descrição, ?estrutura)	Descreve a estrutura dos scan.
Tipos de dados.	temEstrutura(Tipo_Dados, ?estrutura) ^ temValor(Tipo_Dados, ?valor) ^ Expressões(Tipo_Dados) ^ temDescrição(Tipo_Dados, ?descrição) -> sqwrl:select(?descrição, ?valor, ?estru...	Consulta sobre informações de Int, Char, Float.