Ponteiros em Papel: O ensino e a aprendizagem de ponteiros em linguagem de programação C com base em envelopes coloridos

Gustavo Kira[1], **Luiz Ernesto Merkle**[2]

[1]Departamento de Ciência da Computação – Universidade do Estado de Santa Catarina (UDESC) Joinville, SC.

[2]Departamento de Informática – Universidade Tecnológica Federal do Paraná (UTFPR) Curitiba, PR.

gustavokira@gmail.com, merkle@utfpr.edu.br

Abstract. *This paper presents a teaching and learning dynamic with colored envelopes as a way to work with pointers in the C language. Pointer mastery is an important step to understand C at same time it is one of it's most difficult topics. This activity was planned assuming that learning pointers, without a concrete base, privileges those who already have a good algebra understanding or comprehends well the computer's abstract architecture model. Using envelopes enables a concrete analogy and offers an intermediate step for those who still developing their abilities to handle symbolic manipulation or still learning how to deal with the abstract machine model.*

Resumo. *Este artigo apresenta uma dinâmica com envelopes coloridos como um meio para trabalhar o conceito de ponteiro no ensino e aprendizagem da linguagem de programação C. A mestria de ponteiros é um passo importante para o domínio desta linguagem, ao mesmo tempo que é, talvez, um de seus tópicos mais difíceis. Esta atividade foi pensada com o pressuposto de que o a aprendizagem de ponteiros, sem um apoio concreto, privilegia quem já tem um bom domínio de álgebra ou uma boa compreensão abstrata da arquitetura de um computador. O uso de envelopes permite desenvolver uma analogia concreta e oferece um passo intermediário para aqueles que ainda estão por desenvolver suas habilidades ligadas a manipulação simbólica ou a compreensão abstrata da máquina.*

1. Introdução

As dificuldades no ensino e aprendizagem de uma linguagem de programação não são questões novas como já identificou [Aureliano et al. 2016]. Trabalhos como os de [Raiol et al. 2015], [França e Tedesco 2015], [Anido 2015], [Raeder et al. 2016] e [Da Silva et al. 2016] localizam, em especial, o problema na questão da abstração associada à própria ciência da computação. Mesmo que a abstração seja um conceito central tanto para a computação quanto para o campo de seu ensino e aprendizagem, é díficil encontrar uma reflexão um pouco mais profunda sobre o termo. É comum a referência a [Wing 2006] sobre pensamento computacional, mas em nenhum momento é feita uma reflexão sobre o que é a abstração.

Estrutura de dados é uma disciplina consolidada como parte da ciência da computação no Brasil e tem como conteúdos conceitos considerados abstratos e historicamente associados ao desenvolvimento de programas, tais como pilhas: listas e árvores [Nunes et al. 2015]. Nos Estados Unidos este conteúdo é associado aos dois primeiros anos dos cursos de computação (CS1, CS2) [Bucci et al. 2001, Simon et al. 2010]. Mesmo não sendo uma disciplina sobre linguagens de programação, é comum o uso de uma para a implementação dos conceitos associados a disciplina.

No curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS) da Universidade do Estado de Santa Catarina (UDESC), a disciplina de estrutura de dados usa a linguagem C. O conceito de ponteiros desta linguagem é um dos fundamentais para um bom entendimento dos conteúdos da disciplina. Entretanto, também é um dos que discentes têm mais dificuldade [Milne e Rowe 2002, Craig e Petersen 2016].

[Milne e Rowe 2002] também lançam a hipótese de que a dificuldade quanto aos ponteiros e outros conceitos relacionados pode vir da falta de conhecimento, por parte de quem aprende, de como a memória principal do computador funciona, algo também colocado por [Lippert 2018] e por [Craig e Petersen 2016]. Esta hipótese abre espaço para pensar que a habilidade em lidar com conceitos abstratos na computação não é uma característica nata, mas dependente de uma série de outros conceitos do próprio campo.

A observação de [Milne e Rowe 2002] também torna a alta variância de conhecimentos ligados a computação apresentado por quem ingressa em cursos de gradução da área uma variável importante para o ensino e aprendizagem daquilo que é considerado abstrato, pois existe uma certa dependência destes conceitos de outros conceitos da própria computação. O perfil discente das matérias iniciais de um curso de graduação inclui estudantes que estão a mais tempo no curso refazendo a disciplina, egressos de cursos de ensino médio técnico em informática e pessoas que tiveram um contato mais a fundo com a informática pela primeira no ensino superior.

Pensando nessa assimetria de perfis, buscou-se na filosofia e na educação um conceito de abstração que permita lidar um pouco melhor com este problema e que sirva de base para pensar em instrumentos e formas de trabalhar com conceitos abstratos. Com isso, este trabalho propõe enxergar o conceito de abstração como uma forma de mediação ao invés de uma categoria estanque e antagônica ao conceito de concreto. O entendimento como mediação implica também no entendimento do abstrato ou da abstração como sendo sempre um processo que pode ser aprendido e ensinado. Este último desdobramento traz consigo uma perspectiva diferente para a produção de materiais de ensino voltados à computação, uma vez que exercícios, exemplos e exposições devem, de alguma maneira, ajudar os e as estudantes a compreender e obter uma maestria deste processo.

Para iniciar uma discussão sobre esta forma de encarar a abstração, este trabalho está dividido em mais três seções: Primeiro, uma breve introdução das questões que envolvem entender o abstrato como uma mediação entre concretudes; segundo, uma atividade piloto sobre um tema que é tido como abstrato (ponteiros em linguagem C) dentro da computação pensada com o apoio teórico descrito na seção anterior e; por fim, uma pequena discussão sobre o desdobramentos possíveis para a atividade descrita.

2. Abstrato como mediação entre duas concretudes

O minidicionário Aurélio define Abstração como "resultante de abstração", "que opera com qualidades e relações, e não com a realidade" e "que expressa qualidade ou característica separada do objeto a que pertence ou está ligada". Já o concreto, o mesmo dicionário define como "que existe em forma material" e "de consistência mais ou menos sólida" [Holanda 2010].

Para [Meksenas 1992] e [Lefebvre 1991, p.49-50], essa aparente oposição entre os conceitos de concreto e de abstrato é uma espécie de senso comum com raízes na filosofia. A origem do entendimento de ambos como uma oposição binária pode ser localizada na Grécia Antiga, mais precisamente com Aristóteles e a sua distinção entre contemplação, entendida com abstração; e ação, entendida como concretude. A questão torna-se mais complexa do que uma simples associação de ideias ao lembrar que, na Grécia de Aristóteles "todo trabalho prático e produtivo cabia aos escravos, enquanto o pensamento metafísico era uma atividade própria da aristocracia, livre para pensar" [Meksenas 1992, p.93].

Esta perspectiva sobre os conceitos de concreto e de abstrato pode levar a uma visão elitista de conhecimento a qual associa o abstrato a algo somente acessível a uma minoria preparada para tal [Meksenas 1992]. Complementamos que esta mesma perspectiva também leva a entender que existe uma naturalidade para o conhecimento abstrato, algo que a priori não deveria pautar uma agenda de ensino e aprendizagem.

[Silva 2000] também tenta explicar uma outra problemática que surge do uso de oposições binárias como forma de marcação social e linguística: "em uma oposição binária, um dos termos é sempre privilegiado, recebendo um valor positivo, enquanto o outro recebe uma carga negativa" [Silva 2000, p.83]. O positivo é tratado como a "norma" enquanto o negativo é colocado como uma espécie de desvio do que é normal. O problema surge ao entender que esta relação não é simétrica: a norma define o que é correto, muitas vezes percebida como a única forma, delegando ao polo negativo a função de ser "readequado" a norma.

Como pares de conceitos antagônicos e estáticos, tem-se um problema de dualidade: tudo que é concreto não é abstrato, assim como tudo que é abstrato não é concreto. Indo além, se abstração é aquilo que [Silva 2000] chama de "norma", cabe ao concreto a conotação negativa.

[Miliszewska e Tan 2007] mostram uma das problemáticas que deriva deste modelo: "outra dificuldade enfrentada por estudantes de programação está na necessidade de imaginar e compreender vários termos abstratos que não têm equivalentes na vida real[1]". Essa passagem mostra uma percepção que existe um corte claro entre vida real (concreto) e programação de computadores (abstrato), esquecendo das diversas relações entre ambos, desde da presença destes no mundo real até mesmo as analogias possíveis entre os conceitos da vida e do computador.

Para fugir desta dicotomia, [Meksenas 1992] propõe entender o concreto e o abstrato de uma outra forma e apoia-se no entendimento do abstrato como mediação, esta última definida por [Machado 2011, p.55] como algo que situa-se "sempre no meio do

[1] *Another difficulty faced by programming students is the need to imagine and comprehend many abstract terms that do not have equivalents in real life [Miliszewska e Tan 2007]*

processo, constituindo em condição de possibilidade do conhecimento em qualquer área, em vez de ponto de partida ou ponto de chegada. São um degrau necessário que conduz de um patamar de concretude a outro" [Machado 2011, p.55].

[Meksenas 1992, p.95] resume o processo ao colocar que "as diferentes práticas de ensino partem sempre de certos níveis de concretude, chegando, pela abstração, a outros níveis de concretude". Isto faz com que o concreto não seja o contrário do abstrato, nem mesmo podendo ser consideradas entidades de mesmo tipo. Pode-se dizer que o concreto é um dado estado de percepção de um objeto enquanto o abstrato é um processo de passagem para um outro tipo de estado de percepção, este também concreto, mas com outra configuração. A figura 1 ilustra graficamente a proposta de [Meksenas 1992] e torna mais claro qual o papel do abstrato dentro desta perspectiva.

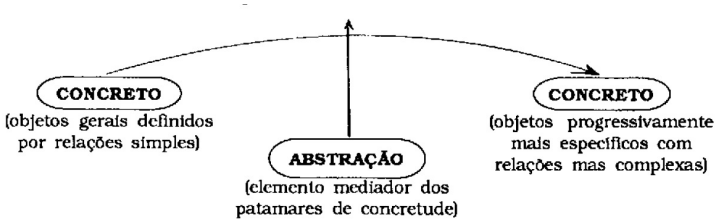

Figura 1. Esquema proposto por [Meksenas 1992]. fonte:[Meksenas 1992, p.95]

[Meksenas 1992] ressalta que o processo não acaba no segundo estágio de "concretude" mas é um processo sem fim. No caso da computação esta perspectiva se alinha muito bem com o tipo de conhecimento exigido na área, dada a grande diversidade e complexidade de temas sob o nome de computação ou informática. Além disso, uma outra implicação mais importante também surge: programação e/ou temas ligados a computação considerados abstratos, na verdade são outros tipos de concretude, mais específicos e mais complexos que os habituais. Ou seja, ensinar e aprender o conceito de árvore apresentado como uma estrutura de dados é fazer uma passagem do conceito de árvore (ser que existe no mundo) para o de árvore (estrutura análoga que possui relações e propriedades ligadas ao computador, a matemática e a planta árvore). Ambos são concretos, mas estão sob óticas diferentes de conhecimento.

Com isso, materiais de ensino para temas tidos como abstratos dentro da computação devem, de alguma maneira, trabalhar com este processo de transformação. Ou seja, dentro desta perspectiva, pode-se lançar a hipótese de que o problema quanto ao aprendizado de abstração é na verdade a falta de medição entre a concretude do corpo discente com a concretude necessária para o domínio dos conceitos a serem trabalhados. Outro entendimento interessante seria o da distância entre as concretudes, uma vez que para um bom entendimento de alguns conceitos, talvez seja necessário diversas transformações de entendimento.

É importante citar que mesmo não usando diretamente o mesmo referencial teórico descrito aqui, existem iniciativas que parecem ter uma mesma premissa, como por exemplo uso de jogos para ensino e aprendizagem de estrutura de dados [Barbosa et al. 2015] e o uso de robótica como apoio ao ensino e aprendizagem de programação [Paparidis e Franco 2016].

Assim, na próxima seção, será apresentada uma atividade que usa envelopes de papel coloridos como uma forma de explicar o funcionamento de ponteiros na linguagem C que não se apoie somente em conceitos da computação (como memória principal ou acesso randômico), mas que use conceitos análogos que fazem parte do cotidiano do corpo discente. Espera-se que este tipo de exercício seja um instrumento de auxílio para que as e os estudantes tenham subsídios para chegar mais perto de um entendimento que inclua os conceitos da computação.

3. Envelopes Coloridos

Ponteiros são um tipo de variável existente na linguagem C e, como colocado anteriormente, são considerados de difícil aprendizado [Milne e Rowe 2002, Adcock et al. 2007, Craig e Petersen 2016]. A ideia de usar envelopes teve como inspiração o experimento dos quatro cartões feito pelo psicólogo inglês Peter Watson em 1966. Para [Machado 2011, 50-53], este experimento mostra que existem duas possibilidades de concretude: uma com relação ao fato do material ser manipulável, uma visão mais próxima do entendimento comum e outra quanto aos significados do material, mais próxima do entendimento de concreto e de abstrato apresentado na última seção. Ao usar envelopes, assim como fez Watson, optou-se por tentar maximizar ambas as dimensões.

Foram desenvolvidos os seguintes materiais: um conjunto de envelopes verdes, um conjunto de envelopes amarelos e um conjunto de envelopes laranjas. Além disso, também foram cortados papéis das mesmas cores para serem colocados dentro dos envelopes. Cada envelope tem duas informações: um "nome" composto por números e letras e um "endereço" composto por apenas números. Na figura 2 temos um envelope de cada cor, sendo o texto grande ("p10", "pp11", "a") o nome do envelope e o número (1005, 2004, 0014) no canto direito inferior o seu endereço.

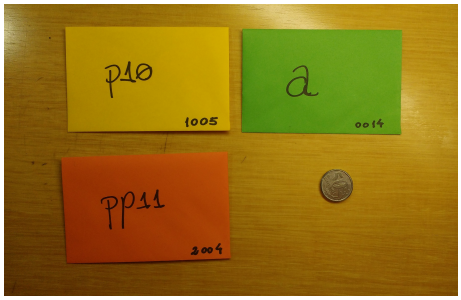

Figura 2. Exemplo dos envelopes usados nesta atividade. fonte: próprio autor.

Já os papéis, dependendo da sua cor, podem ter valores diferentes. Os verdes sempre têm valores numéricos, os amarelos têm somente números equivalentes aos possíveis endereços dos envelopes verdes e os laranjas têm somente números equivalentes aos possíveis endereços dos envelopes amarelos. A figura 3 mostra um exemplo no qual o envelope de nome "pp11" tem um papel com valor 1005 que é o mesmo valor do endereço do envelope amarelo "p10" que por sua vez tem um papel com valor 0014, endereço do envelope verde.

A relação com o conceito de ponteiros é dada da seguinte maneira: Os envelopes verdes representam as variáveis de um programa escrito em C. Tem um nome, tem um endereço de memória e podem ter um valor. O valor é o papel verde guardado dentro

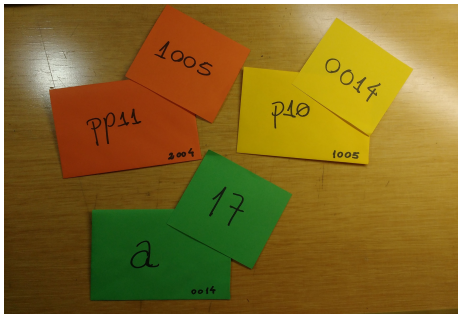

Figura 3. Exemplo dos envelopes e papéis correspondentes. fonte: próprio autor.

do envelope. Os envelopes amarelos são ponteiros de envelope verde, já que guardam somente valores que podem ser endereços de envelopes verdes. E por fim, os envelopes laranjas são ponteiros de envelope amarelo, mantendo a mesma relação vista entre os amarelos e verdes. A figura 4 ilustra um exemplo no qual o envelope amarelo é um ponteiro para o envelope verde, pois tem dentro de si o valor do endereço do envelope verde de nome "a". Ou seja, os envelopes verdes são similares as variáveis inteiras, os amarelos a ponteiros para inteiros e os laranjas são ponteiros para ponteiros de inteiros.

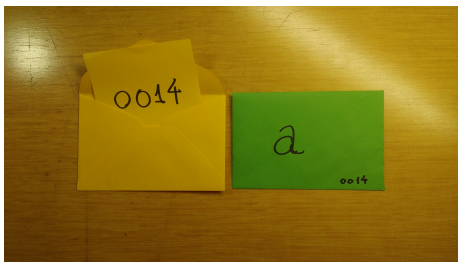

Figura 4. Exemplo de como funciona a analogia com o conceito de ponteiro. fonte: próprio autor.

Foram apresentadas também três regras de manipulação dos envelopes. Primeiro, é possível abrir um envelope chamando-o pelo seu nome. Segundo, é possível pedir o endereço de um envelope usando o sinal de (&) + (o nome do envelope). Terceiro, é possível usar o sinal (*) + (nome do envelope) para ler o valor de um envelope cujo endereço está dentro daquele no qual foi aplicado o sinal de (*). É importante notar que estas são operações associadas ao uso de ponteiros na linguagem C.

Foram pensadas três possibilidades de uso para este conjunto de materiais. (a) Como apoio a uma explicação teórica, (b) como uma atividade interativa, (c) como uma espécie de jogo. Somente as duas primeiras foram executadas, pois durante o planejamento do jogo, este foi desmembrado em um outro projeto com seus próprios materiais. Entretanto, isto não impede de que um jogo usando os envelopes seja feito, apenas que, para esta iteração ele foi descartado.

3.1. Explicação Teórica Concreta

A técnica descrita neste artigo foi aplicada na UDESC, em uma disciplina de estrutura de dados do curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS). Na

grade atual, esta disciplina encontra-se no terceiro semestre do curso. Dado o contexto, todos os e as estudantes que participaram da atividade têm o ensino médio completo e algum tipo de experiência com a linguagem C.

A atividade foi feita com o objetivo de revisar o conceito de ponteiros, pois é um conceito importante para a aprendizagem da implementação das estruturas de dados em C. Ela também foi uma resposta a dificuldade de uma parte considerável da turma em resolver alguns exercícios com ponteiros passados uma semana antes. Ainda cabe mencionar que junto dos discentes do TADS, também frequentam a disciplina estudantes do bacharelado em Ciência da Computação da mesma instituição.

O primeiro uso dos envelopes e cartões foi como material de apoio para uma revisão expositiva sobre ponteiros em C. Neste momento foi feita a apresentação do que são, e como funcionaria a lógica dos envelopes descrita anteriormente. Como o corpo discente já teve pelo menos uma disciplina de linguagem C, as associações entre os envelopes e ponteiros foram também apontadas neste momento.

3.2. Ponteiros em Papel

Explicado como funciona os envelopes, foi pedido para que cada um da turma pegasse um envelope e um valor de mesma cor. De posse deste material, cada estudante deveria cumprir instruções como as seguintes: 1) Escreva uma instrução que execute uma soma com 2 envelopes verdes e 1 amarelo. 2) Escreva uma instrução que execute uma soma usando 3 envelopes amarelos. 3) Escreva uma instrução com apenas envelopes amarelos que execute uma troca do valor de um envelope verde. 4) É possível guardar um papel amarelo em um envelope laranja ou vice-versa?

Uma vez apresentada as questões, os e as estudantes deveriam procurar colegas que tenham outros envelopes e usar as regras dos operadores (*) e (&) para tentar achar envelopes que os ajudariam a responder às questões. A figura 5 mostra a forma como um estudante respondeu as instruções. Neste caso, em especial, é interessante notar que foi feita uma substituição de termos equivalentes a cada passo.

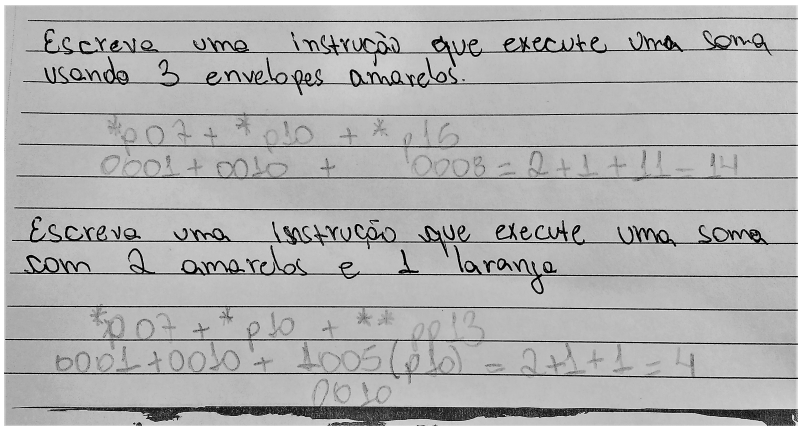

Figura 5. Exemplo de resposta do exercício. fonte: próprio autor.

Vale a pena citar que nem todos prestaram atenção quando foi dito para que pegassem somente os papéis e envelopes separados para a atividade. Alguns pegaram materiais que estavam em uma outra pilha que não deveria ser usada. Para resolver este problema,

alguns estudantes pegaram vários envelopes e organizaram de um modo que fosse possível responder as instruções. Este desvio também se mostrou uma forma contingente para explicar o que acontece ao tentar usar uma variável não declarada.

3.3. Da Concretude do Papel para a Concretude da Tela

Além das atividades com os envelopes, também foi feita uma outra atividade que consistia em transpor a lógica dos envelopes para um código na linguagem C. Foi dado um código inicial com três variáveis: a primeira chamada envelopeVerde e do tipo inteiro(int) com um valor qualquer, a segunda chamada envelopeAmarelo do tipo ponteiro para inteiro com o endereço da variável envelopeVerde com seu valor e a terceira chamada envelopeLaranja do tipo ponteiro para ponteiro de inteiro com o endereço da variável envelopeAmarelo como seu valor. Com isso, o exercício pedia para que as e os estudantes completassem o trecho de código para que ao ser executado fosse exibido na tela: o valor e o endereço de cada uma das variáveis o que obriga os estudantes a aplicarem os operadores (*) e (&), também alvos da atividade com envelopes coloridos.

Esta passagem é importante para que fique clara a relação entre a atividade prática e o conteúdo da disciplina. No código 1, temos um exemplo de trecho de código produzido na atividade. Ele foi alterado para poder ser reproduzido, entretanto as mudanças não afetam a essência do código.

```
1
2     int envelopeVerde = 17;
3     int* envelopeAmarelo = &envelopeVerde;
4     int** envelopeLaranja = &envelopeAmarelo;
5
6     //Parte 1
7     printf("O valor do envelope verde: %i\n", envelopeVerde);
8     printf("O valor do endereco do envelope verde: %i\n", &envelopeVerde);
9
10    //Parte 2
11    printf("O valor do envelope amarelo: %p\n",*envelopeAmarelo);
12    printf("O valor do endereco do envelope Amarelo: %i\n", &envelopeAmarelo);
13    printf("O valor contido no endereco do envelope Amarelo:\n", envelopeAmarelo);
```

Código 1. Exemplo de trecho de código produzido na atividade.

4. Considerações Finais

Mesmo que executada somente uma vez, os códigos entregues e as respostas dos estudantes durante as atividades foram boas o suficiente para dar continuidade ao trabalho sobre esta técnica. Neste momento, são imaginadas quatro frentes: 1) Refinamento do material (envelopes, papéis e disposição das informações). 2) Aplicação em grupos com perfis diferentes 3) Desenvolvimento de mais atividades que usem os mesmos materiais e 4) Novas aplicações desta mesma atividade com apoio de instrumentos de pesquisa.

Quanto ao primeiro tópico, é possível ir em duas direções: Para mais perto ou mais longe do computador. Mais perto, seria usar os endereços de uma forma mais parecida com os endereços da memória principal de um computador. Por exemplo, separar uma quantidade de memória de acordo com o tipo da variável e fazer com que os endereços nos envelopes respeitem estes espaços. Por outro lado, é também seria possível usar endereços de ruas ou casas para trabalhar os conceitos de ponteiros de uma forma semelhante a forma como trabalha a computação desplugada.

Ainda sobre os materiais, também é possível criar arquivos digitais para que o material seja reproduzido por qualquer um que tenha interesse. Para concretizar este passo será preciso estudar uma licença aberta adequada e preparar o material para ser distribuído como um recurso educacional aberto (REA).

Sobre a aplicação em grupos com perfis diferentes, uma primeira possibilidade está em estudantes de ensino médio frequentadores de cursos técnicos integrados ligados as disciplinas de informática e/ou computação. Esta restrição a um grupo específico de secundarista se dá pelo próprio conceito de ponteiros, muito mais útil e proveitoso para quem estuda computação ou informática.

Como a atividade descrita foi aplicada com o objetivo de revisão, também seria interessante ver o seu desempenho ao ser usada como introdução ao conceito de ponteiro da linguagem C.

Por fim, também seria muito rico aplicar novamente esta mesma atividade, mas com apoio de questionários ou outras ferramentas de medição a fim de ter resultados controlados e passíveis de comparação com outras abordagens.

Referências

Adcock, B., Bucci, P., Heym, W. D., Hollingsworth, J. E., Long, T., e Weide, B. W. (2007). Which pointer errors do students make? In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 9–13, New York, NY, USA. ACM.

Anido, R. (2015). Saci - ainda outro ambiente para o ensino de programação. In *Anais do 35º Congresso da Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15)*.

Aureliano, V. C. O., Tedesco, P. C. d. A. R., e Giraffa, L. M. M. (2016). Desafios e oportunidades aos processos de ensino e de aprendizagem de programação para iniciantes. In *Anais do 36º Congresso da Sociedade Brasileira da Computação (CSBC'16), 24º Workshop sobre Educação em Computação (WEI'16)*.

Barbosa, W. A., Nunes, I. d. F., Inocêncio, A. C. G., Oliveira, T. B. d., e Júnior, P. A. P. (2015). Deg4trees:um jogo educacional digital de apoio ao ensino de estrutura de dados. In *Anais do 35º Congresso da Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15)*.

Bucci, P., Long, T. J., e Weide, B. W. (2001). Do we really teach abstraction? *SIGCSE Bull.*, 33(1):26–30.

Craig, M. e Petersen, A. (2016). Student difficulties with pointer concepts in c. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '16, pages 8:1–8:10, New York, NY, USA. ACM.

Da Silva, L. M. P., Bonfim, B. C., Silva, R. C., e da Silva, J. B. (2016). Poogame: Um jogo sériopar ao ensino de programação orientada a objetos. In *Anais do 36º Congresso da Sociedade Brasileira da Computação (CSBC'16), 24º Workshop sobre Educação em Computação (WEI'16)*.

França, R. S. d. e Tedesco, P. (2015). Explorando o pensamento computacional no ensino médio: do design à avaliação de jogos digitais. In *Anais do 35º Congresso da*

Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15).

Holanda, A. B. d. (2010). *Mini Dicionario Aurelio Lingua Portuguesa*. Positivo livros.

Lefebvre, H. (1991). *Lógica formal, lógica dialéctica*. Civilização Brasileira, Rio de Janeiro.

Lippert, E. (2018). References are not addresses. disponível em <https://blogs.msdn.microsoft.com/ericlippert/2009/02/17/references-are-not-addresses>. Último acesso em 31 de mar. de 2018.

Machado, N. J. (2011). *Matemática e língua materna*. Cortez: Autores Associados, São Paulo.

Meksenas, P. (1992). As noções de concreto e abstrato: sua relação com as práticas de ensino. *Revista da Faculdade de Educação*, 18(1):92–98.

Miliszewska, I. e Tan, G. (2007). Befriending computer programming: A proposed approach to teaching introductory programming. In *Issues in Informing Science and Information Technology*.

Milne, I. e Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information Technologies*, 7(1):55–66.

Nunes, M. M., Costa, E. B. B., Feitosa, F. A., e Brancher, J. D. (2015). Análise das ementas de estruturas de dados das universidades brasileiras. In *Anais do 35º Congresso da Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15)*.

Paparidis, O. S. e Franco, M. E. (2016). Plataforma arduino como apoio ao ensino de programação no curso de técnico em informática integrado. In *Anais do 36º Congresso da Sociedade Brasileira da Computação (CSBC'16), 24º Workshop sobre Educação em Computação (WEI'16)*.

Raeder, M., Py, M., Rigo, S., e Pinheiro, J. (2016). L2pm: relato de uma experiência sobre o ensino integrado de lógica, programação e matemática para computação. In *Anais do 36º Congresso da Sociedade Brasileira da Computação (CSBC'16), 24º Workshop sobre Educação em Computação (WEI'16)*.

Raiol, A. A. C., Sarger, J., Souza, A., Sivaldo, S., e Fábio, B. (2015). Resgatando a linguagem de programação logo: Uma experiência com calouros no ensino superior. In *Anais do 35º Congresso da Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15)*.

Silva, T. T. d. (2000). A produção social da identidade e da diferença. In da Silva, T. T., editor, *Identidade e diferença: a perspectiva dos estudos culturais*, chapter 2, pages 73–102. Editora Vozes, Rio de Janeiro.

Simon, B., Clancy, M., McCartney, R., Morrison, B., Richards, B., e Sanders, K. (2010). Making sense of data structures exams. In *Proceedings of the Sixth International Workshop on Computing Education Research*, ICER '10, pages 97–106, New York, NY, USA. ACM.

Wing, J. M. (2006). Computational thinking. *Commun. ACM*, 49(3):33–35.