Capítulo 14

Desenvolver o Pensamento Computacional Usando Seguir e Dar Instruções

José Figueiredo, Francisco José García-Peñalvo

***Title*— Improving Computational Thinking Using Follow and Give Instructions**

***Abstract* — Computational Thinking can be defined as a set of skills for problem solving based on Computer Science. Young people grow up surrounded by technology but many of them go for university without any prior knowledge in computer science. This paper is an attempt to demonstrate the importance of computational thinking in the first beginning of learning programming, and what activities best contribute to increase the abilities of each computer engineering student in computational thinking according to the characteristics of those who attend the Polytechnic of Guarda, Portugal. Most of our students have never had the opportunity to learn computational thinking.**

***Keywords*— computer science, computer science education, programming.**

***Resumo*— O Pensamento Computacional pode ser definido como um conjunto competências mentais e técnicas, baseadas nas ciências da computação, para a resolução de problemas. Os jovens de hoje crescem rodeados de tecnologia, no entanto, chegam ao ensino superior com poucos conhecimentos na área das ciências da computação e com o pensamento computacional pouco desenvolvido. Neste trabalho procuramos demonstrar a importância do pensamento computacional nos primeiros anos de aprendizagem da programação de computadores, e quais as atividades que melhor contribuem para o desenvolvimento dessas competências, de acordo com as caraterísticas dos alunos que iniciam os estudos do curso de engenharia informática, no Instituto Politécnico da Guarda, Portugal. A maioria dos nossos alunos nunca teve a oportunidade de aprender pensamento computacional.**

***Keywords*— ciências da computação, ensino ciências da computação, programação.**

I. Pensamento Computacional - Visão Geral

O pensamento Computacional é uma aptidão fundamental para todos. O termo Computational Thinking, ou Pensamento Computacional, tornou-se popular pela Jeannete M. Wing [1] . A autora defende a disseminação massiva do pensamento computacional, tal como a leitura, a escrita e a aritmética. Nos últimos anos, temos assistido à proliferação de projetos, com o apoio de entidades governamentais e não-governamentais, como o objetivo especifico de incentivar o pensamento computacional e a aprendizagem da programação, em especial nos primeiros anos de escolaridade.

O pensamento computacional é uma aptidão que nos permite criar soluções para problemas utilizando as técnicas da computação [2], [3], [4]. Os computadores são usados para nos ajudar a resolver problemas. No entanto, para que tal seja possível é necessário conhecer bem o problema e identificar, analisar e implementar possíveis soluções. O pensamento computacional dá-nos os métodos, as técnicas e a coragem para o examinar, avaliar e criar possíveis soluções para um problema complexo. As soluções podem ser apresentadas de forma que os computadores, os humanos, ou ambos, as possam entender. Para a maioria das nossas tarefas do dia-a-dia, das mais simples às mais complexas, é uma boa ideia fazer um plano de resolução utilizando técnicas das ciências da computação, como: dividir o problema complexo em problemas mais simples de fácil compreensão e resolução – decomposição; procurar semelhanças com outras experiências e soluções – reconhecer padrões; centrar a atenção apenas no que é essencial e expressar uma solução em termos genéricos para que possa ser utilizada em outras situações – abstração; desenvolver um conjunto de instruções passo-a-passo que conduzam à resolução do problema – algoritmo. Este plano pode ser utilizado por todos, independentemente da sua área de conhecimento, tarefa ou idade.

O pensamento computacional é essencial para o desenvolvimento de aplicações informáticas, mas também pode ser utilizado para apoiar a resolução de problemas pelas

Este trabalho foi apresentado originalmente em Fifth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'17)

José Figueiredo, Unidade de Investigação para o Desenvolvimento do Interior Av. Dr. Francisco de Sá Carneiro, 50 6300-559, Guarda, Portugal Telef +351271220191. Fax +351271220150; e-mail: jfig@ipg.pt.

Francisco José García-Peñalvo Computer Science Department Research Institute for Educational Sciences GRIAL research group University of Salamanca fgarcia@usal.es

Alfonso Lago Ferreiro, André Fidalgo, Oliveira da Silva
TICAI 2017 TICs para el Aprendizaje de la Ingeniería. ISBN 978-84-8158-774-6

mais diversas áreas da ciência, desde das matemáticas às humanidades. Os estudantes que aprendem pensamento computacional ao longo da sua vida começam a ver com mais clareza o relacionamento entre os diferentes assuntos, bem como as relações entre a escola e a vida fora da sala de aula.

O pensamento computacional ensina-nos a pensar, a encontrar a solução para um problema, a organizar um plano de resolução de uma tarefa. O pensamento computacional ensina-nos, também, as atitudes e predisposições necessárias à confiança, à capacidade e à persistência em lidar com a complexidade dos problemas.

Neste artigo destacamos a importância da aprendizagem e do treino do pensamento computacional ao longo da vida de todos os estudantes. De facto, a maioria dos estudantes que chega à universidade nunca teve a oportunidade de desenvolver as competências e técnicas do pensamento computacional. A programação é uma das melhores formas de desenvolver essas competências. Na primeira parte do nosso trabalho apresentamos algumas das ferramentas que ajudam a motivar e a incentivar o gosto pela programação, especialmente aquelas que disponibilizam recursos e ajudam os professores. Na segunda parte do nosso trabalho, apresentamos os resultados de um estudo envolvendo um grupo de estudantes de engenharia informática do Instituto Politécnico da Guarda, Portugal. Neste estudo, exploramos os conceitos de seguir e dar instruções e desenho de mapas, numa tentativa de melhorar e desenvolver as competências do pensamento computacional de cada aluno.

II. Programação o Caminho para o Pensamento Computacional - Ferramentas

Ciência, Tecnologia, Engenharia e Matemáticas (STEM - Science, Technology, Engineering and Mathematics) estão entre as principais características da uma vida moderna e possuem a chave para resolver muitos dos desafios atuais e futuros mais urgentes da humanidade. A posição dos Estados Unidos na economia global está em declínio, em parte porque os trabalhadores dos EUA não possuem conhecimento fundamental nestas áreas. Para estudar o declínio da competitividade dos EUA e melhorar a preparação dos trabalhadores nestas áreas, a Framework for K-12 Science Education propõem uma nova abordagem para o sistema de ensino básico e secundário, que permitirá captar o interesse dos estudantes e aumentar as suas qualificações nestas áreas [5]. Este é apenas um exemplo, muitos outros países sentem a necessidade de promover e incentivar o trabalho nesta área para melhorar os conhecimentos nas áreas STEM, principalmente desenvolvendo ferramentas para que todos os envolvidos na educação tenham um trabalho facilitado, como é caso do Reino Unido [6].

É conhecido que em 2020 haverá mais 1.4 milhões de vagas de emprego na área da informática, mas de acordo com as atuais previsões, as pessoas com qualificações para ocupar essas vagas é de aproximadamente 30% [7]. O emprego na área da informática e nas tecnologias da informação e comunicação deverá crescer de 12% entre 2014 e 2024, representando o mais rápido crescimento que a média de todas as outras áreas de emprego, segundo U.S. Bureau of Labor Statistics, Office of Occupational Statistics and Employment Projections. Por outro lado, existem vários benefícios com a inclusão das ciências da computação nos programas do ensino k-12 (em Portugal representa o ensino básico e secundário) [7]. Estes incluem, entre outros, a construção de competências de nível elevado do pensamento e o aumento de uma atitude positiva perante a área das ciências da computação e informática.

Apesar de toda a dedicação e esforço generalizado na área das ciências da computação e do pensamento computacional, na sua maioria os professores não possuem qualificações profissionais nestas áreas. Esta é provavelmente a razão pela tal, nos últimos anos, testemunhamos à proliferação de inúmeros projetos com o objetivo específico de encorajar o estudo da programação, especialmente nos anos de ensino pré-universitários. Muitas organizações estão a trabalhar arduamente com o objetivo de proporcionar aos jovens o seu sucesso no mundo digital. No trabalho apresentado em [8], foi realizada uma pesquisa exaustiva e respetiva avaliação das ferramentas existentes para o ensino e aprendizagem da programação no ensino pré-universitário. De seguida, apresentamos algumas dessas ferramentas e adicionamos outras, especialmente aquelas que permitem aos professores e pais mostrar aos alunos o melhor caminho:

- **Alice**. Alice é um inovador ambiente de programação baseado em blocos que facilita a criação de animações, a construção de narrativas interativas, ou a programação de jogos simples em 3D. Alice foi desenvolvida para ensinar lógica e pensamento computacional, os princípios fundamentais da programação e, ainda, ser o primeiro contacto com a programação orientada pelos objetos. O projeto Alice fornece ferramentas e materiais adicionais para o ensino, e com resultados comprovados em cativar diferentes grupos de alunos desmotivados [9].
- **Barefoot**. Barefoot disponibiliza um conjunto de recursos que permite aos professores do ensino básico ganhar a confiança, o conhecimento e as competências necessárias para ensinar ciências da computação. Recursos de acordo com o curriculum para todos os países do Reino Unido. Estes recursos incluem planos de aula e *workshops*, todos com o objetivo de ajudar os professores a ganhar confiança e levar as ciências da computação à sala de aula [10]. A *Barefoot Computing* é apoiada pela *British telecommunications* (BT) [11] para ajudar a construir uma cultura de alfabetização tecnológica e computação na escola (CAS – Computing At School) [12].
- **Blockly**. Blockly é uma biblioteca à qual é adicionado um editor de código visual para a construção de aplicativos web e android. O editor Blockly usa blocos gráficos interligados para representar conceitos de código, como variáveis, expressões lógicas, ciclos e muitos outros. Blockly permite que os utilizadores apliquem os princípios da programação sem a preocupação de erros de sintaxe ou a intimidação de um cursor a piscar numa linha de comando. Blockly, criado pela Google, desenvolveu uma grande série de recursos, programas, bolsas de estudo e oportunidades de concessão e financiamento com o intuito de envolver estudantes e professores interessados em desenvolver as suas competências em ciências da computação [13].
- **CodeCombat**. CodeCombat é uma plataforma para aprender ciências da computação enquanto se joga um jogo real. CodeCombat não usa blocos, pois são da

opinião que apresentar código escrito aos estudantes é fundamental para aprendizagem da programação. Permite, também, maior flexibilidade e criatividade na resolução de problemas. CodeCombat é dotado de um conjunto de recursos e guias dos cursos dirigidos aos professores, capacitando-os para o processo de democratização da aprendizagem da programação [14]

- **Code.org**. Lançado em 2013, o Code.org é uma organização sem fins lucrativos dedicada em divulgar e ampliar o conhecimento em ciências da computação e, também, aumentar e incentivar a participação das mulheres. O seu principal objetivo é de que cada aluno tenha a oportunidade de aprender ciências da computação e a programação de computadores. Defendem que as ciências da computação e programação de computadores deve ser parte fundamental de qualquer curriculum de ensino, juntamente com outros cursos de ciência, tecnologia, engenharia e matemáticas (STEM), tal como biologia, física, química e álgebra. A "Hora do Código" é uma iniciativa global da *Computer Science Education Week* [15] e Code.org com o intuito de dar a conhecer e incentivar a programação de computadores.
- **Cubetto**. Cubetto é inspirado em Logo Turtle. Cubetto é um robô de fácil aprendizagem e utilização, construído em madeira, que ensina os conceitos básicos da programação de computadores. Utiliza uma linguagem de programação muito simples que pode ser tocada e manipulada como o Lego. É adequada para crianças que ainda não sabem ler ou escrever [16].
- **CS Unplugged**. Os seus princípios são: não é necessário computador; ciências da computação real; aprender fazendo; diversão; não é necessário equipamento específico; incentivar variações; para todos; atividades cooperativas; individual; flexível [17].
- **Greenfoot**. Greenfoot ensina programação orientado pelos objetos em Java. Greenfoot é visual e interativo. Funciona com atores que são programados em código java, proporcionando um experiência com a programação tradicional de escrita de código com a execução visual. Greenroom é um local exclusivo a professores para partilha de recursos didáticos e de discussão em redor do ensino e aprendizagem do Greenfoot [18].
- **Khan Academy**. Khan Academy oferece um conjunto de exercícios práticos, vídeo educativos e um plano de aprendizagem personalizado que permite a cada aluno, aprender e estudar ao seu próprio ritmo, dentro e fora da sala de aula, nas mais diversas áreas: matemáticas, ciências, engenharia, informática, artes, humanidades, economia e finanças. A sua missão é proporcionar uma educação gratuita para todos, em qualquer lugar [19].
- **Kodable**. Escrito por professores do pré-escolar, o curriculum Kodable proporciona a aprendizagem da programação para crianças entre os 4 e 10 anos de idade, mas é especialmente dirigido às crianças do pré-escolar. Kodable possibilita atividades práticas em grupo ou individuais e permite desenvolver a criatividade, a comunicação e a colaboração. O seu objetivo é alcançar o maior número de alunos e que as ciências da computação façam parte da educação no pré-escolar [20].
- **Kodu**. Kodu permite aos alunos criar jogos para computador, em sistema Windows, através de uma linguagem simples de programação visual. Kodu pode ser explorado para a ensinar e desenvolver a criatividade, a resolução de problemas, a narração de histórias, bem como a programação. Qualquer um pode usar o Kodu para fazer um jogo, desde as crianças mais pequenas aos adultos sem qualquer experiência ou conhecimento em programação [21].
- **Lightbot**. Com o Lightbot os alunos devem orientar um robô para iluminar um conjunto de blocos. Para tal, o aluno deve programar o robô utilizando um conjunto de instruções. O jogo é multiplataforma. O objetivo principal é compreender como criar e dar ao computador um conjunto de instruções a seguir [22].
- **LiveCode**. LiveCode tem a visão de que todos podem codificar. Criaram uma plataforma de código aberto para criar aplicações multiplataforma. A linguagem LiveCode foi projetada para ser expressiva, legível, memorável e o mais próximo possível da forma como falamos e pensamos. O ambiente de programação visual permite ao utilizador desenvolver aplicações numa linguagem simples e poderosa, possibilitando aos alunos desenvolver as suas competências em ciências da computação [23].
- **MIT App Inventor**. MIT App Inventor é um ambiente intuito de programação visual, que permite a todos a criar aplicações para dispositivos móveis, como smartphones e tablets. O projeto MIT App Inventor. Procura democratizar o desenvolvimento de software, capacitando todas as pessoas, especialmente os jovens, para a transição de consumidores de tecnologia para os criadores de tecnologia [24]. App Inventor for Educators é uma comunidade educativa, como o objetivo de partilhar ideias, recursos, perguntas e respostas [25].
- **Scratch**. Scratch é uma linguagem de programação e uma comunidade *online* onde todos podem programar e partilhar as suas experiências, como histórias, jogos e animações, com pessoas de todo o mundo. À medida que as crianças "brincam" com o Scratch, elas aprendem a pensar de uma forma criativa, trabalham de um modo colaborativo, e questionam sistematicamente. Estas são algumas das caraterísticas fundamentais para a vida no século XXI. O Scratch foi desenvolvido e é gerido pelo Lifelong Kindergarten group, do MIT Media Lab [26].
- **Snap** (Build Your Own Blocks). Snap é uma linguagem de programação visual *drag-and-drop*. É um complemento ao Scratch, a principal diferença é que Snap permite criar novos blocos, de acordo com as nossas necessidades, e desta forma criar programas mais complexos. Esta caraterística adicional torna-o mais adequado para um nível de programação mais avançado [27].
- **TACCLE 3.** TACCLE 3 – Coding é um Projeto Europeu [28] de apoio ao ensino primário e todos os professores que desejam ensinar informática aos alunos com idades entre os 4 e os 14 anos. TACCLE 3 é um projeto que fornece conhecimento e ideias práticas que os professores podem usar e os materiais de que precisam para introduzir a informática ou a codificação nas suas salas de aula [29]–[31].

- **Touch Develop**. Touch Develop permite criar aplicações para computador ou dispositivos móveis num ambiente de programação bastante agradável. Podemos, também, partilhar as nossas aplicações na cloud. Disponibiliza um curriculum (Creative Coding Through Games And Apps) para a introdução às ciências da computação utilizando Touch Develop. Este curriculum inclui planos diários para implementar com diferentes tempos de duração [32].
- **Tynker**. Tynker é uma forma revolucionária de aprender a programar. As crianças aprendem os conceitos fundamentais da programação com blocos visuais, progredindo depois para Javascript e Python à medida que desenvolvem os seus conhecimentos e ganham confiança nas suas capacidades [33].

III. A Nossa Proposta

Os projetos de massificação do pensamento computacional e da codificação, neste trabalho por vezes também designado por ciências da computação com um sentido mais lato, começa a dar os primeiros passos para a sua implementação no sistema de ensino em Portugal. Esta é a principal razão pela qual a maioria dos estudantes, que chegam ao ensino superior, nunca tiveram a oportunidade de aprender ou desenvolver o pensamento computacional e a codificação.

Neste trabalho, foi realizado um estudo para investigar e explorar as opiniões e dificuldades que os alunos enfrentam quando iniciam o estudo de um curso de programação. Nesta pesquisa é, também, nossa intenção detetar qual ou quais as melhores práticas para desenvolver e melhorar as competências dos alunos na aprendizagem da programação.

Os jovens, os nossos alunos, crescem rodeados de tecnologia. Mas, na sua maioria não fazem a mais pequena ideia de como essa tecnologia funciona e o quanto isso é importante para o seu futuro. Os jovens de hoje em dia não conhecem a vida sem a tecnologia. É parte integrante da sua existência. A sua maioria passa todos os tempos livres ao computador ou com dispositivos móveis. São ferramentas essenciais de comunicação e informação para eles. Crescem com os computadores e telemóveis. Por tudo isto, procuramos não usar a tecnologia para este estudo. É intencional que os alunos manipulem e resolvam os exercícios manualmente, como um jogo de tabuleiro, onde os alunos possam explorar com prazer, manualmente, sem medo de errar e onde o relacionamento professor-aluno e a confiança possam ser melhorados e aperfeiçoados.

A. *Grupo de Estudo*

Este estudo envolve um grupo de alunos (46) do curso de engenharia informática, do Instituto Politécnico da Guarda (IPG), Portugal. O nosso grupo de estudo revela algumas dificuldades gerais na área das ciências da computação, conclusão obtida através de um inquérito inicial, sobre os seus conhecimentos e competências adquiridas ao longo dos seus anos de estudo no ensino pré-universitário.

B. *Curso de Pré-Programação*

O pensamento computacional pode ser aplicado a vários tipos de problemas que não envolvem diretamente tarefas de codificação. De acordo com as características dos alunos de engenharia informática do IPG, criamos um conjunto de exercícios, ou atividades, para que os alunos possam desenvolver e melhorar substancialmente as suas capacidades cognitivas. E, consequentemente melhorar a aprendizagem da programação, a que nos designamos de curso de pré-programacão [34].

O curso engloba o seguinte conjunto de atividades:

- Seguir e Dar instruções. Este tipo de exercícios tem como objetivo aumentar o desenvolvimento das aptidões de raciocínio cognitivo e visualização espacial dos alunos, características fortemente associadas às competências necessárias para a programação [35]–[37]. Neste tipo de exercícios os alunos devem desenhar numa folha de papel o que um aluno ou o professor descreve por palavras. Em outro exemplo, os alunos devem descrever direções, ou seja, o que tem de fazer se deslocar de um ponto A para o ponto B.
- Desenho de mapas. Com este tipo de atividade procuramos desenvolver no aluno as suas capacidades de planeamento, descrição e abstração perante uma situação real. Estas atividades incluem o desenho de mapas e indicação do caminho a seguir para atingir um objetivo específico.
- Origami e dobrar papel. Origami é uma arte secular japonesa difundida em todo mundo [36], [38]–[40]. Conhecida pelo desenvolvimento de características como: perceção visual e espacial, coordenação motora fina, memória, paciência e persistência, autoconfiança, pensamento lógico, atenção, concentração e alívio do stress e tensão. Existem milhares de exemplos, do mais simples ao mais complexo, de várias categorias, que podem ser usados de acordo com as características e gosto de cada aluno, como podemos ver na .
- Dobrar papel, em particular dobrar e perfurar o papel, é frequentemente usado para aferir das capacidades de visualização espacial das pessoas. No nosso caso, usamos esta atividade para aferir, desenvolver e melhorar as capacidades de visualização espacial do aluno. Neste tipo de atividade, os alunos devem imaginar que estão a dobrar papel e perfurar esse papel dobrado. De seguida devem imaginar que desdobram o papel e devem identificar a posição dos furos no papel, ver Figura 2.
- Memory Transfer Language (MTL) é um conjunto de atividades de manipulação de instruções simples de um

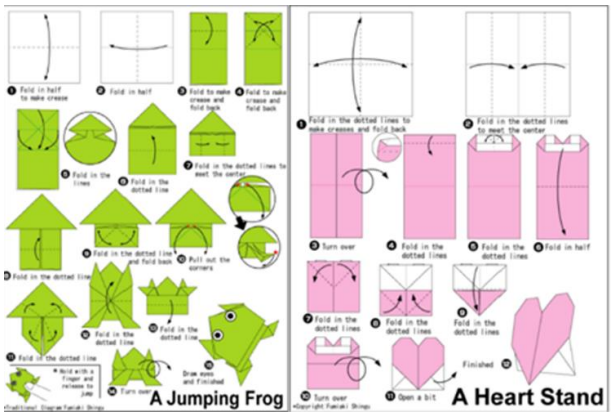

Figura .1. Exemplo de origami: a jumping frog, a heart stand.

programa e representa-las na memória do computador. Este tipo de atividade permite superar alguns problemas detetados na construção do conhecimento na aprendizagem da programação, particularmente na representação de variáveis e instruções de atribuição. A metodologia utilizada para implementar este tipo de atividade baseia-se numa tabela, ver , representando a memória do computador, onde o aluno vai representado passo a passo as instruções, ver Figura , de um programa simples [41].

- Parson Problems são atividades usadas na aprendizagem da programação, onde o aluno deve selecionar e organizar de um conjunto de instruções, as instruções para completar um determinado objetivo [42]–[44].

C. Descrição e Resultados do Estudo.

Foram desenvolvidos duas atividades. Uma atividade com 20 exercícios de papel dobrado e perfurado (PH – Punched Holes) e outro com 5 exercícios de seguir e dar instruções (FGI – Follow and Give Instructions). Efetuaram estas atividades 46 alunos. A atividade PH foi avaliada pelo número de respostas corretas. A atividade de FGI foi classificada de acordo com uma escala de valor zero, para os alunos que não responderam, até a um valor de quatro, para os alunos que responderam corretamente com detalhe e utilizando referências espaciais. Na TABELA I podemos observar os resultados da primeira atividade PH.

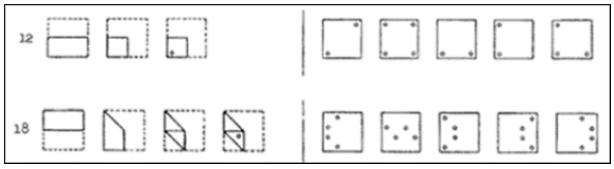

Figura .2. Exemplo Papel dobrado perfurado, adaptado de [39].

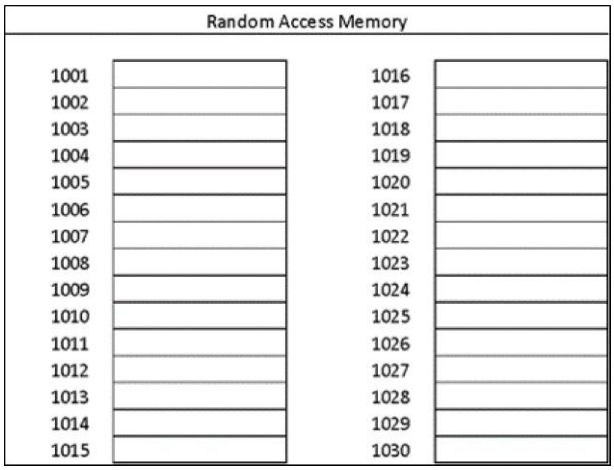

Figura .3. Grelha para representação das instruções das atividades MTL.

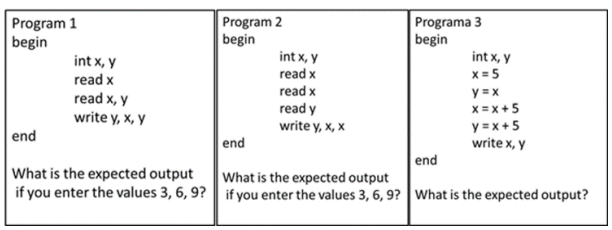

Figura .4. Exemplos de programas para atividades MTL.

Na primeira linha da tabela, PH < 10, 9 dos 46 alunos obtiveram um resultado inferior a 10 respostas corretas. Dos 9 alunos, 6 concluíram a unidade curricular de introdução à programação, do curso de engenharia informática, com sucesso.

Como podemos facilmente verificar na **¡Error! No se encuentra el origen de la referencia.**, 20 alunos obtiveram um resultado de FGI >= 3, desses alunos 18 completaram com sucesso a unidade curricular de introdução à programação, o que corresponde a uma percentagem de 90%.

Na TABELA III, apresentamos os resultados combinados para as duas atividades. É de salientar os resultados obtidos para a atividade FGI >= 3. Os resultados são de 100%, 88,9% e 100% para os diferentes resultados de PH. Com estes resultados estamos em querer que a atividade de seguir e dar instruções (FGI - Follow and Give Instructions) tem uma forte influência no sucesso da unidade curricular de introdução à programação. Pois dos 35 alunos que foram avaliados com FGI >= 3, 33 obtiveram sucesso na unidade curricular de introdução à programação, o que corresponde uma percentagem de 94,2%.

IV. PRATICAR PENSAMENTO COMPUTACIONAL

Estudos demonstraram a relação entre o estilo e o nível de detalhe na descrição e construção de mapas com os objetivos de um curso de programação [35] . De acordo com os nossos resultados, também podemos afirmar que existe uma forte relação entre o estilo e o nível de detalhe usado na descrição das instruções a seguir, quer no desenho de objetos e mapas, ou seja nas atividades de FGI, como os objetivos da unidade curricular de introdução à programação, do curso de engenharia informática do IPG.

TABELA I
RESULTADOS DA ATIVIDADE PH.

		Success	%
PH < 10	9	6	66,7%
PH >= 10	37	25	67,6%
PH >= 14	27	21	77,8%

TABELA II
RESULTADOS DA ATIVIDADE FGI.

		Success	%
FGI < 3	26	13	50,0%
FGI >= 3	20	18	90,0%

TABELA III
RESULTADOS COMBINADOS PH E FGI.

		Success	%
PH < 10 and FGI < 3	7	4	57,1%
PH < 10 and FGI >= 3	2	2	100,0%
PH >= 10 and FGI < 3	19	9	47,4%
PH >= 10 and FGI >= 3	18	16	88,9%
PH >= 14 and FGI < 3	10	6	60,0%
PH >= 14 and FGI >= 3	15	15	100,0%

A. *Praticar Desenho de Mapas e Seguir e Dar Instruções*

Com base na nossa experiência, pretendemos implementar um conjunto de exercícios para trabalhar e estudar esta metodologia, com o intuito de melhorar as competências dos alunos em pensamento computacional e programação. Nestas atividades, serão avaliados o nível de detalhe e a clareza na descrição que conduz à resolução do problema. Alguns exemplos:

- Seguir e dar instruções, ou perguntar e dar direções de localização, como a metodologia usada nos cursos de línguas, onde com base em mapas fornecidos aos alunos, como os da Figura e da Figura , os alunos devem descrever detalhadamente o caminho necessário para se deslocar de um ponto A para um ponto B.
- Seguir e dar instruções também deve incluir atividades em que os alunos devem desenhar, numa folha de papel, de acordo com as instruções dadas por outro aluno, ou pelo professor, como os exemplos da Figura e da Figura .

B. *Aprender pela Experiência de Ensinar*

Os benefícios de utilizar a experiência de ensinar para aprender são amplamente conhecidas. As pessoas aprendem mais e melhor se tiverem a necessidade de ensinar a outros, sugerem os autores do trabalho realizado em [45]. Em função destes resultados, vamos implementar atividades onde os alunos devem preparar uma atividade, lição, para outros alunos, sejam eles os próprios colegas ou alunos de outras escolas e de idades inferiores. Esta atividades devem ter por objetivo o desenvolvimento do pensamento computacional. Estas atividades podem ter também como objetivo a participação na "Hora do Código". Estas atividades podem, também, ser utilizadas como um meio de divulgação e promoção do pensamento computacional e programação pelas escolas da região da Guarda, Portugal, realizadas pelos alunos com a supervisão dos professores.

V. Conclusão

Neste artigo, apresentamos algumas ferramentas que desenvolvem e melhoram as competências do pensamento computacional. No entanto, essas ferramentas devem ser utilizadas ao longo de todo o processo de aprendizagem, elas devem ser manipuladas e exploradas desde os 3 anos de idade aos 18 anos, a idade em que habitualmente os alunos ingressam no ensino superior [46], [47]. O sistema de ensino em Portugal começa agora a dar os primeiros passos na implementação do pensamento computacional como componente curricular ou extracurricular. Como é o caso da programação e robótica no ensino básico [48]. A maioria dos alunos do primeiro ano do curso de engenharia informática, do IPG, nunca teve a oportunidade de desenvolver o pensamento computacional ao longo da sua vida estudantil. Assim sendo, as dificuldades de aprendizagem nos primeiros anos nas unidades curriculares de programação são uma preocupação constante. É necessário trabalhar esses alunos de uma forma rápida e efetiva, antes que a falta de interesse e motivação tome conta destes alunos.

Com este trabalho, pretendemos apresentar a nossa ideia e divulgar os resultados obtidos com as nossas atividades de seguir e dar instruções. Achamos os resultados bastante interessantes e, portanto, devem ser explorados. Como tal, sugerimos um conjunto de atividades para testar com os alunos. Existem muitas estratégias que os professores podem utilizar para aumentar as competências dos estudantes com dificuldades na aprendizagem nas ciências da computação. Sendo educação computacional e, especialmente o pensamento computacional, uma área relativamente nova,

Figura .5. Mapa 1 a utilizar em exercícios de dar direções.

Figura .6. Mapa 2 a utilizar em exercícios de dar direções.

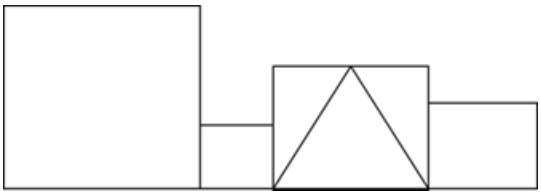

Figura .7. Imagem a utilizar em atividades de seguir e dar instruções.

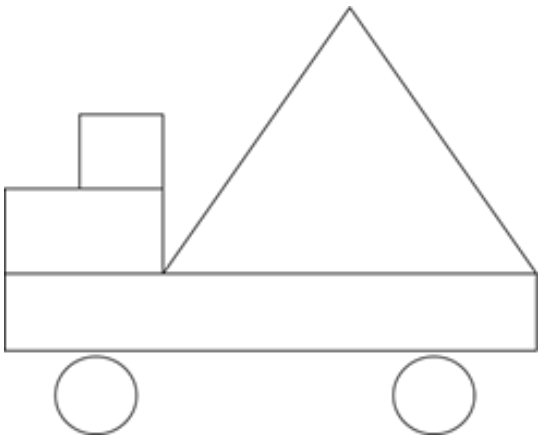

Figura .8. Imagem a utilizar em atividades de seguir e dar instruções.

quando comparada com áreas como as matemáticas, física ou línguas, muitos dos educadores não sabem como lidar e o que oferecer para apoiar os alunos com mais dificuldades de aprendizagem na área das ciências da computação. Neste artigo, sugerimos algumas estratégias e recursos que os professores podem implementar para apoiar os seus alunos.

REFERENCES

[1] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, p. 33, Mar. 2006.

[2] F. J. García-Peñalvo, "What Computational Thinking Is," *J. Inf. Technol. Res.*, vol. 9, no. 93, 2016.

[3] F. J. García-Peñalvo and J. Cruz-Benito, "Computational thinking in pre-university education," in *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM '16*, 2016, pp. 13–17.

[4] F. J. García-Peñalvo and A. J. Mendes, "Exploring the computational thinking effects in pre-university education," *Comput. Human Behav.*, Dec. 2017.

[5] *A Framework for K-12 Science Education*. Washington, D.C.: National Academies Press, 2012.

[6] "STEM." [Online]. Available: https://www.stem.org.uk/. [Accessed: 06-Jul-2017].

[7] M. Israel, Q. M. Wherfel, J. Pearson, S. Shehab, and T. Tapia, "Empowering K-12 Students with Disabilities to Learn Computational Thinking and Computer Programming," vol. 48, no. 1 OP-TEACHING Exceptional Children, v48 n1 p45-53 Sep-Oct 2015. 9 , p. 45, 2015.

[8] F. J. García-Peñalvo, J. Hughes, A. Rees, I. Jormanainen, T. Toivonen, D. Reimann, M. Tuul, and M. Virnes, "Evaluation Of Existing Resources (Study/Analysis)," Jan. 2016.

[9] "Alice – Tell Stories. Build Games. Learn to Program." [Online]. Available: http://www.alice.org/. [Accessed: 05-Jul-2017].

[10] "Home - Barefoot Computing Barefoot Computing." [Online]. Available: https://barefootcas.org.uk/. [Accessed: 05-Jul-2017].

[11] "techliteracy." [Online]. Available: https://techliteracy.co.uk/. [Accessed: 06-Jul-2017].

[12] "Computing At School." [Online]. Available: http://www.computingatschool.org.uk/. [Accessed: 06-Jul-2017].

[13] "Blockly | Google Developers." [Online]. Available: https://developers.google.com/blockly/. [Accessed: 06-Jul-2017].

[14] "CodeCombat - Learn how to code by playing a game." [Online]. Available: https://codecombat.com/. [Accessed: 05-Jul-2017].

[15] "Computer Science Education Week." [Online]. Available: https://csedweek.org/. [Accessed: 05-Jul-2017].

[16] "Cubetto: A robot teaching kids code & computer programming." [Online]. Available: https://www.primotoys.com/. [Accessed: 06-Aug-2017].

[17] "Computer Science Unplugged." [Online]. Available: http://csunplugged.org/. [Accessed: 05-Jul-2017].

[18] "Greenfoot | About Greenfoot." [Online]. Available: https://www.greenfoot.org/overview. [Accessed: 07-Jul-2017].

[19] "Khan Academy | Free Online Courses, Lessons & Practice." [Online]. Available: https://www.khanacademy.org/. [Accessed: 05-Jul-2017].

[20] "Programming for Kids | Kodable." [Online]. Available: https://www.kodable.com/. [Accessed: 05-Jul-2017].

[21] Microsoft Research - FuseLabs, "Kodu | Home." [Online]. Available: https://www.kodugamelab.com/. [Accessed: 21-Dec-2017].

[22] "Lightbot." [Online]. Available: https://lightbot.com/flash.html. [Accessed: 05-Jul-2017].

[23] LiveCode Ltd, "LiveCode in Education | LiveCode." [Online]. Available: https://livecode.com/products/livecode-platform/livecode-in-education/. [Accessed: 06-Jul-2017].

[24] "MIT App Inventor." [Online]. Available: http://ai2.appinventor.mit.edu/. [Accessed: 05-Jul-2017].

[25] "App Inventor for Educators – MIT App Inventor Educators Community." [Online]. Available: http://teach.appinventor.mit.edu/. [Accessed: 06-Jul-2017].

[26] "Scratch - Imagine, Program, Share." [Online]. Available: https://scratch.mit.edu/. [Accessed: 05-Jul-2017].

[27] "Snap! (Build Your Own Blocks) 4.0." [Online]. Available: http://snap.berkeley.edu/index.html. [Accessed: 10-Jul-2017].

[28] "Taccle 3 – Supporting primary teachers to teach coding." [Online]. Available: http://www.taccle3.eu/en/. [Accessed: 05-Jul-2017].

[29] F. J. García-peñalvo, "A brief introduction to TACCLE 3 – Coding European Project," pp. 3–6, 2016.

[30] F. J. García-Peñalvo, D. Reimann, M. Tuul, A. Rees, and I. Jormanainen, "TACCLE 3, O5: An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers KA2 project " TACCLE 3 – Coding " (2015-1-BE02-KA201-012307)," in *TACCLE3 Consortium*, 2016, p. 72.

[31] F. J. García-Peñalvo, "Proyecto TACCLE3 – Coding," in *Ediciones Universidad de Salamanca - XVIII Simposio Internacional de Informática Educativa.*, 2016, pp. 187–189.

[32] "Microsoft Touch Develop - create apps everywhere, on all your devices!" [Online]. Available: https://www.touchdevelop.com/. [Accessed: 06-Jul-2017].

[33] "Coding for Kids | Tynker." [Online]. Available: https://www.tynker.com/. [Accessed: 05-Jul-2017].

[34] J. Figueiredo, N. Gomes, and F. J. García-Peñalvo, "Ne-course for learning programming," in *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM '16*, 2016, pp. 549–553.

[35] S. Fincher, B. Baker, I. Box, Q. Cutts, M. De Raadt, P. Haden, J. Hamer, R. Lister, M. Petre, A. Robins, K. Sutton, D. Tolhurst, and J. Tutty, "Computer Science at Kent programming courses," no. 1, 2005.

[36] Simon, S. Fincher, A. Robins, B. Baker, I. Box, Q. Cutts, M. De Raadt, P. Haden, J. Hamer, M. Hamilton, R. Lister, M. Petre, K. Sutton, D. Tolhurst, and J. Tutty, "Predictors of success in a first programming course," *Proc. 8th Austalian Conf. Comput. Educ. - Vol. 52*, pp. 189–196, 2006.

[37] N. E. Study, "An Overview of Tests of Cognitive Spatial Ability," *66th EDGD Mid-Year Conf. Proc.*, p. 6, 2012.

[38] Z. Falomir, "Towards A Qualitative Descriptor for Paper Folding Reasonin," in *Proc. of the 29th International Workshop on Qualitative Reasoning (QR'16)*, 2016.

[39] A. J. Jaeger, J. Wiley, J. Pellegrino, K. Zinsser, M. Stieff, and T. Moher, "What Does the Punched Holes Task Measure?," 2015.

[40] S. Cooper, K. Wang, M. Israni, and S. Sorby, "Spatial Skills Training in Introductory Computing," *Proc. Elev. Annu. Int. Conf. Int. Comput. Educ. Res.*, pp. 13–20, 2015.

[41] L. J. Mselle and H. Twaakyondo, "The impact of Memory Transfer Language (MTL) on reducing misconceptions in teaching programming to novices," *Int. J. Mach. Learn. Appl.*, vol. 1, no. 1, pp. 1–6, May 2012.

[42] B. J. Ericson, "Adaptive Parsons Problems with Discourse Rules," *Icer '14*, pp. 145–146, 2014.

[43] P. Denny, A. Luxton-Reilly, and B. Simon, "Evaluating a new exam question: Parsons problems," *Proc. fourth Int. Work. Comput. Educ. Res.*, pp. 113–124, 2008.

[44] B. B. Morrison, L. E. Margulieux, B. Ericson, and M. Guzdial, "Subgoals Help Students Solve Parsons Problems," *Proc. 47th ACM Tech. Symp. Comput. Sci. Educ.*, pp. 42–47, 2016.

[45] J. F. Nestojko, D. C. Bui, N. Kornell, and E. L. Bjork, "Expecting to teach enhances learning and organization of knowledge in free recall of text passages," *Mem. Cognit.*, vol. 42, no. 7, pp. 1038–1048, 2014.

[46] F. J. García-Peñalvo, F. L. Largo, X. M. Prieto, and E. Vendrell, "Educación en Informática sub 18 (EI<18)," *ReVisión*, pp. 13–18.

[47] F. L. Largo, F. J. García-Peñalvo, X. M. Prieto, and E. V. Vidal, "La enseñanza de la informática, la programación y el pensamiento computacional en los estudios preuniversitarios," in *Education in the Knowledge Society*, 2017, pp. 7–17.

[48] ERTE - Equipa de Recursos e Tecnologias Educativas - Ministério da Educação, "Programação e Robótica no Ensino Básico | ERTE." [Online]. Available: http://erte.dge.mec.pt/programacao-e-robotica-no-ensino-basico-0. [Accessed: 21-Dec-2017].

José Alberto Quitério Figueiredo licenciado em Engenharia Informática pelo Instituto Politécnico da Guarda, Portugal. Possui mestrado em Engenharia Electrotécnica e de Computadores, pela Faculdade de Engenharia da Universidade do Porto, Portugal. Atualmente é doutorando em Engenharia Informática, no Departamento de Informática y Automática da Facultad de Ciencias, da Universidad de Salamanca, Espanha. É docente na Escola Superior de Tecnologia e Gestão, do Instituto Politécnico da Guarda, Portugal, desde 1992, na área de Programação e Multimédia.

Francisco José García Peñalvo bacharel em computação pela Universidade de Salamanca, Espanha, e pela Universidade de Valladolid, Espanha. Doutorado pela Universidade de Salamanca, Espanha. Atualmente é o Diretor do grupo de investigação Research Group in Interaction and e-Learning (GRIAL). As suas principias áreas de investigação centram-se em e-Learning, computers and education, adaptive systems, web engineering, semantic web, e reutilização de software. Liderou e participou em mais de 50 projetos de investigação e inovação. Foi Vice-Chanceler Technological Innovation da Universidade de Salamanca, entre Março de 2007 e Dezembro de 2009. Publicou mais de 200 artigos em revistas e conferências internacionais. Foi editor convidado em várias edições especiais de revistas internacionais (Online Information Review, Computers in Human Behavior, and Interactive Learning Environments). É o editor chefe da revista Education in the Knowledge Society, e de the Journal of Information Technology Research. É o coordenador do programa de doutoramento em Educação da Sociedade do Conhecimento, da Universidade de Salamanca.