International Journal for Innovation Education and Research

ONLINE ISSN: 2411-2933 PRINT - ISSN: 2411-3123

# Introduction of Basic Computer Programming Logic in Elementary and High Schools Using Logic Block

Ronaldo Lopes Gomes;Ricardo Chamy do Nascimento;Caio Victor da Silva Viga;Bruno Pereira Gonçalves;Jean Mark Lobo de Oliveira;Rilmar Pereira Gomes;David Barbosa de Alencar

## Abstract

This document analyzes factors that make it difficult for computational programming logic to consolidate itself as a subject to be taught in schools and how to introduce it into the academic environment through a hypothetical curriculum based on studies and also including the logical block programming language, Scratch. Studies are conducted to create the basis of an experimental school grade. Said grade describes in detail the age groups it will reach and what knowledge will be taught to them. With a better understanding of programming and programming logic in mind, not only individuals who choose to work in the field will already have a larger knowledge base than today, but people who choose not to work in the field of computer technology will benefit of a more logical and cohesive line of reasoning.

**Keyword:** Computational programming logic; Elementary and high schools; Experimental curriculum; Logic block programming language; Line of reasoning.

**Published Date**: 11/30/2019 **Page.732-742** **Vol 7 No 11 2019**

**DOI**: https://doi.org/10.31686/ijier.Vol7.Iss11.1928

# Introduction of Basic Computer Programming Logic in Elementary and High Schools Using Logic Block

**Ronaldo Lopes Gomes**
ronald42void@gmail.com
Centro Universitário Metropolitano de Manaus – FAMETRO

**Ricardo Chamy do Nascimento**
rkchamy@hotmail.com
Centro Universitário Metropolitano de Manaus – FAMETRO

**Caio Victor da Silva Viga**
vitor.da.alvorada@gmail.com
Centro Universitário Metropolitano de Manaus – FAMETRO

**Bruno Pereira Gonçalves (Advisor)**
goncalves.bruno@gmail.com
Centro Universitário Metropolitano de Manaus – FAMETRO

**Jean Mark Lobo de Oliveira**
jeanlobolive@gmail.com
Centro Universitário Metropolitano de Manaus – FAMETRO

**Rilmar Pereira Gomes**
rilmargomes@hotmail.com
Centro Universitário Metropolitano de Manaus – FAMETRO

**David Barbosa de Alencar**
david002870@hotmail.com
Centro Universitário Metropolitano de Manaus – FAMETRO

## Abstract

*This document analyzes factors that make it difficult for computational programming logic to consolidate itself as a subject to be taught in schools and how to introduce it into the academic environment through a hypothetical curriculum based on studies and also including the logical block programming language, Scratch. Studies are conducted to create the basis of an experimental school grade. Said grade describes in detail the age groups it will reach and what knowledge will be taught to them. With a better understanding of programming and programming logic in mind, not only individuals who choose to work in the field will already have a larger knowledge base than today, but people who choose not to work in*

*the field of computer technology will benefit of a more logical and cohesive line of reasoning.*

**Keywords:** Computational programming logic; Elementary and high schools; Experimental curriculum; Logic block programming language; Line of reasoning.

## 1. Introduction

Computers and technology everywhere in today's world, a very visible fact, and quite obvious in recent decades since the breakthroughs of technology initiated by the World War II technology race and, later, the Cold War space race, the later presenting mankind with great achievements, ranging from wireless (Wi-Fi) connection to the countless orbiting satellites of our planet. Ironically, even with its obvious importance to today's highly technological world, the area of programming is still obscure and relatively far from the reach of younger minds, specifically high school students, ie individuals that are choosing the path they want to take in the job market, leaving the programming area to be discovered by the most curious minds and taught by online portals and forums.

Simplified programming teaching is far from a recent topic: In 1968, Wally Feurzeig, Seymour Papert, and Cynthia Solomon created the programming language known as Logo, which is entirely devoted to education, famous for its use of a cartoon figure of a turtle, where users enter commands to move said turtle around the screen, being quite simple and accessible to almost every type of user, and even with such simplicity, it is still considered as a sophisticated language (PRADO, 2000).

Not only there are tools to learn from, most of the public approves the teaching of programming to the younger generation, as an example, former US President Barack Obama commented about his support programming in schools because, in addition to being important to everyone's future in general, it is important to the country's future (referring to the United States), and adds: "Don't just buy a video game, make one. Not only download the latest app, help develop it. Don't just play on your phone, schedule it!"(OBAMA, 2013). As we can see, there is certainly fresh soil for programming on the educational fields, with public's support and many tools to use, but sadly, from a realistic point of view, it's "easier said than done".

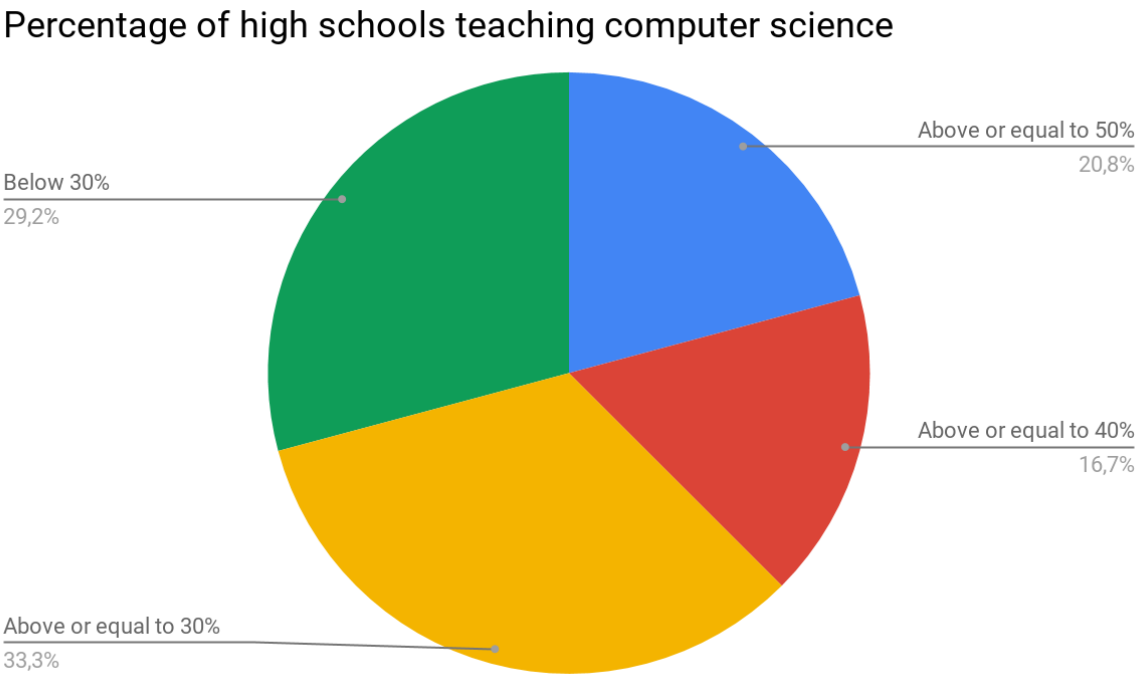
Recently, in 2018, a study performed by Code.org Advocacy Coalition and the Computer Science Teachers Association (CSTA), collected data from the United States of America's progress towards the implementation of Computer Science on high schools, the result being slightly worrying.

Table 1: Public High Schools Teaching Computer Science (CS): Overall, by Community, by Underrepresented Minority, by Free and Reduced Lunch

| | Overall Implementation | By Community Type | | By Percent Underrepresented Minorities | | By Percent Free and Reduced Lunch | |
|---|---|---|---|---|---|---|---|
| State | Percent Teaching CS | Urban | Rural | Under 50% URM | Over 50% URM | Under 50% FRL | Over 50% FRL |
| Alabama | 27% | 49% | 32% | 38% | 42% | 57% | 30% |
| Arkansas | 65% | 67% | 68% | 69% | 61% | 67% | 68% |
| California | 32% | 35% | 28% | 44% | 31% | 48% | 30% |
| Florida | 19% | 23% | 15% | 24% | 21% | 22% | 23% |
| Georgia | 49% | 47% | 45% | 58% | 42% | 79% | 39% |
| Indiana | 51% | 56% | 50% | 55% | 49% | 58% | 46% |
| Iowa | 45% | 53% | 42% | 47% | 56% | 49% | 38% |
| Kansas | 23% | 38% | 14% | 22% | 38% | 26% | 18% |
| Kentucky | 33% | 34% | 35% | 35% | 33% | 46% | 30% |
| Louisiana | 16% | 19% | 9% | 18% | 14% | 31% | 10% |
| Massachusetts | 49% | 51% | 41% | 55% | 36% | 59% | 35% |
| Mississippi | 18% | 24% | 29% | 34% | 21% | 39% | 26% |
| Missouri | 30% | 37% | 26% | 33% | 40% | 41% | 27% |
| Montana | 40% | 57% | 37% | 43% | 22% | 46% | 27% |
| New York | 34% | 33% | 29% | 42% | 26% | 44% | 27% |
| North Carolina | 39% | 40% | 37% | 42% | 35% | 48% | 30% |
| North Dakota | 22% | 43% | 18% | 24% | 8% | 24% | 14% |
| Oklahoma | 24% | 34% | 18% | 23% | 29% | 27% | 23% |
| Oregon | 30% | 35% | 20% | 33% | 22% | 41% | 26% |
| Rhode Island | 78% | 76% | 88% | 76% | 81% | 81% | 76% |
| South Carolina | 39% | 47% | 40% | 51% | 35% | 56% | 36% |
| Utah | 54% | 58% | 39% | 58% | 35% | 59% | 45% |
| Virginia | 59% | 66% | 53% | 65% | 51% | 65% | 52% |
| Wisconsin | 32% | 37% | 30% | 37% | 21% | 40% | 23% |

Source: Code.org, 2018 State of Computer Science Education. (2018).

The graphic above represents the data collected from almost all high schools from 24 north american states, and worrying results set in, as it becomes clear that the amount of high schools that do teach computer science barely reaches 50% on most states




Graphic 1: Percentage of high schools teaching computer science, based on table 1.
Source: Made by author, data from Table 1.

On Graphic 1, made for simplicity's sake, shows that barely ¼ of the north american states on the study conducted by Code.org have more than 50% of their high schools teaching computer sciences on site, and while only half of the north american states were studied, this data can picture that, while general public certainly has approved the idea, as said before, "it's easier said than done".

## 2. Methodology

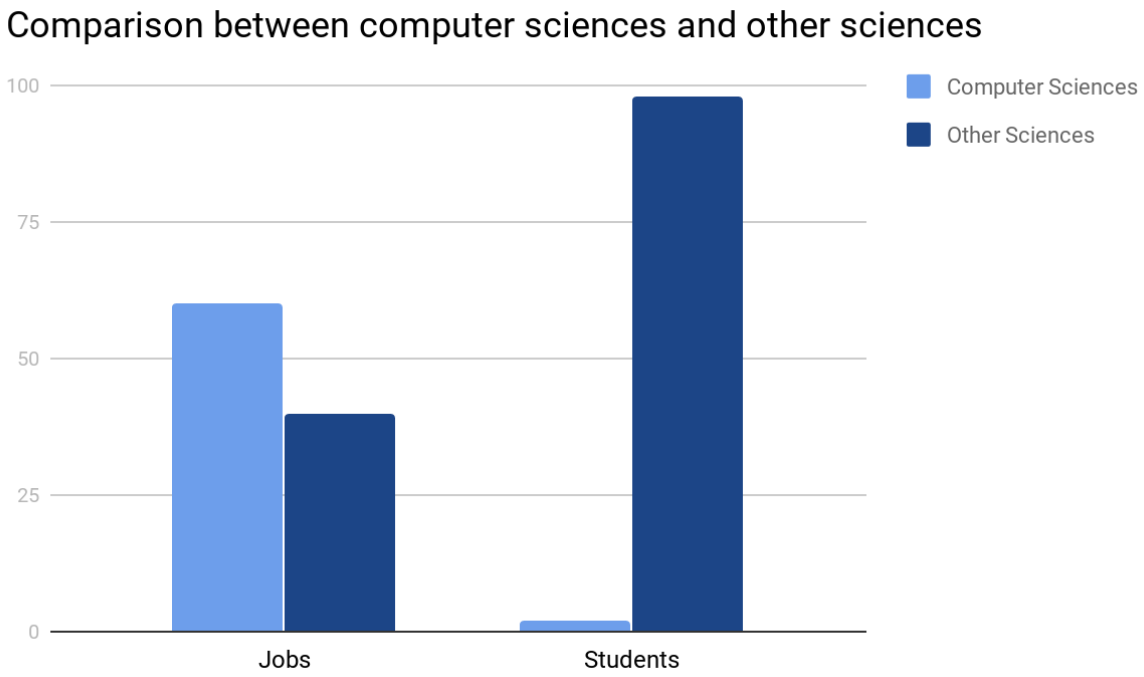
In the study conducted, apart from general concepts and arguments taken from a variety of sources, concepts and bases were acquired from the current "experiment" carried out in the United Kingdom, which imposes programming as a compulsory subject in schools, and the appropriate academic grid that it applies. In order to meet the need for a block language that is intuitive and easily accessible and easy to use, it's used the tool widely used in teaching programming in the pedagogical environment called Scratch.

## 3. Development

### *3.1 Theoretical Reference*

#### *3.1.1 Why Teach?*

It is of popular knowledge that today's society has computational technology as an extremely important base, not only for entertainment, but also for economic and social purposes. In view of this, the labor market has expanded according to the technological evolutions of recent decades, opening up countless opportunities for those interested in the area, while obscuring them for those who were not even aware of such opportunities.




Graphic 2: Comparison between Computer sciences and other sciences on the professional market.
Source: Made by Author, based on code.org's data.

In 2016, the famous social network Linkedin posted on its blog a list of "10 skills you will be hired for in 2017", citing data and cloud computing as dominant on the list, as well as statistical analysis and data mining, which are in high demand for being cutting-edge technology skills, with contractors having a need to have employees with such skills to stay in the market competition (IGNATOVA, 2016), reinforcing the point that failing to present the computing world as a school subject results in a huge shortage of skilled professionals in the field of work, simply because no one has ever received encouragement or even knowledge of the field and its possibilities.

In addition to the financial benefits that come with teaching computer science, teaching concepts of computer programming logic would greatly benefit individuals with a more logical and cohesive line of thinking, even if they choose to work in other areas of the labor market.

A point to be emphasized would be, as mentioned earlier, the greater support and interest that has been developing in the public for programming, providing the teaching of programming logic with greater acceptance by the masses both students and their parents. In 2004, Thai students offered programming classes to 81 students from 3 elementary schools, featuring 3 different software over a 2-week period. At the end of the experiment, it was found that 43 of the parents (78.18%) support teaching programming languages in schools, while 50 parents (90.9%) said their children spent more than an hour working on programming activities at home, and some kids spent even whole afternoons trying to add more sophisticated features and aspects to their programs. (LIN, et al 2004).

Finally, one of the most important aspects of today's software would be ease of use. We are all familiar with the classic black screen with white or green letters, where the programmer enters each command line individually without any indication of what will or will not happen, this classic stereotype reinforced by

movies and TV is no longer as realistic as you might think. With the many advancements and upgrades that today's software introduces, many already have intuitive, easily accessible interfaces, informing the user of bugs, misused components, missing packages, going so far as to even introduce interface building to the program being developed automatically, requiring only screen assembly by dragging and placing components with the mouse while the software itself takes care of the coding process.

### *3.1.2 How to Teach the Teacher?*

The most obvious question in this equation is the first step in the great transition presented by the proposal, it should be clear that the teacher is not there, initially at least, to teach and explain lines of codes and how they work, the main point should be to teach kids programming logic, as in, teaching them how to think in a most logical and cohesive way, usually common to professionals of the computational area. In short: if a person knows the basics of programming logic, they can understand almost every programming language. The next topic at hand is: how to prepare teachers? Let us take the following example: in 2013, the UK officially included computer programming in its curriculum, serving as a "guinea pig" in an ambitious experiment to introduce public elementary and high school youth to programming, and even before the experiment came into play. One of the biggest problems was already under consideration: how to teach teachers how to teach programming, which led the British government to raise about 1.1 million euros in December 2013 for the teaching of primary school teachers, and about 500,000 euros in 2014 to attract business to help train teachers (DREDGE, 2014). The point here is: such a monumental move obviously won't be cheap, since the teacher themselves suddenly will have a need to know how the basic concepts of programming logic, how to apply it and, most importantly/obviously, how to properly teach it to younger minds in a way that, not only the students fully grasp the nature of said way of thinking, but also that said logic can evolve along with the student along the years, ever improving and ever helping said individual to take more logical and calculated decisions.

### *3.1.3 Teaching Programming Logic*

Obviously, trying to teach elementary school children how to program in Java or C ++ even before teaching them multiplication is not a good idea, so at this stage of school life we focus on teaching how to think logically. With that in mind, comes the question: How do you teach computer programming logic? Sophie Deen, leader of the nonprofit group known as Code Club Pro, said in an interview with pri.org that "the biggest change is teaching [computer science] to elementary school kids, [...] sounds harder than which actually is. An algorithm is a simple step-by-step instruction for solving any particular problem", using for example a teacher named Phillip Bagge, who in turn used the procedure of "assembling" a jelly sandwich, with the teacher being the "computer", and challenging his students to "program" him to assemble the sandwich. Deen also said that "It's actually quite easy and fun to teach algorithms to 5-year-olds, and make them start thinking that computers are stupid [...]" (WOOLF, 2014). The use of logic in general is a factor to be considered by all, not only computer professionals (MANZANO & DE OLIVEIRA, 2005), therefore, when the child realizes that one of the most important aspects Programming is understanding how computer programming logic works and how to apply it not only in programming, but in your everyday life, this opens up the potential of the individual to learn to program in virtually any and every programming

language.

With the logical line of reasoning formed in the early years of teaching the younger individual, the next step is to apply that line of reasoning practically by introducing the basic programming concepts themselves to the student, which theoretically would have no problem assimilating. Basic commands and lines of code, as well as what they do and how they affect the line of code execution, as the most basic commands (IF, WHERE, WHEN, and others), besides being self-explanatory and easy to remember, have had its aspects broken and assimilated by the previously formed line of logical thinking of students that has been constantly evolving over the years, and even with the constant social stigma that "programming is hard, left only to professionals and super intelligent people", keep in mind that with the constant evolution of programming and the way an individual programs, in addition to creating new simpler interfaces, programming is no longer an impossible and extremely difficult thing to do as some people still think, programming becomes something simple, resulting in less effort to develop a project in all its aspects (Armstrong, et al 1996). However, even with such simplicity, prudence in such an experiment is more than advisable, so let us consider the following question: how to introduce programming for young people who already have a logical line of thought without overloading them?

### *3.1.4 Structured Language*

Also known as Structured Programming, structured language is a programming paradigm that focuses on repetition structures, subroutines, and other related factors, being the dominant paradigm in software writing for a long time, being succeeded by object-oriented programming. However, even though it is not as widely used in the professional field as it used to be, structured programming has never fallen out of use since it is often one of the primary means by which people learn basic programming concepts due to its general simplicity. This simplicity in certain aspects is an extremely important factor when it comes to teaching not only specific programming languages, but also the programming logic and algorithm concepts themselves for individuals who will delve into the intricacies of programming as a whole, not just restricted to structured programming, mainly because, as said earlier, most people learn the basics from it, using said knowledge to push onward into the vast and fascinating world of programming.

### *3.1.5 Logical Block Programming*

Usually refereed to as "Visual programming language" (VPL), logical block programming languages present themselves to the user with a friendly, simple and easy to assimilate interface, where commands and basic concepts of programming are represented with colorful blocks/panels, properly and well identified, always focusing on ease of use and help the user to meet his goal, giving said user the possibility of programming by a visual and graphical interface instead of the traditional method of typing each and every command line, said work is left to the machine itself, which writes every line of code on the background, leaving an intuitive and easy to use safe zone for the user to learn from, experiment and apply the basics of programming logic early discussed.

### *3.1.6 Scratch in the Educational Environment*

Well known in the programming community for young people, Scratch, as defined on its own website, "is

a programming language and online community in which children can program and share interactive multimedia, such as stories, games and animations, with people from all over the world"(Scratch).

As stated earlier, young people would arrive in Scratch's environment with logical thinking in mind and developed since its incorporation into elementary school, and by introducing such a tool for these young people, they would learn how to apply said logic in a simpler, intuitive and safe environment, being free to explore and evolve their programming logic without having to worry about mistakes, errors, or most importantly, walls of texts and lines of codes.

Not only it is already used by many individuals who already teach programming basics to younger minds nowadays, it is also extremely friendly to younger minds with cartoonist graphics, possibilities to create fun little animations and mini-games, and of course, without leaving the programming part int the dark, presenting itself in a very easy to learn/understand way.

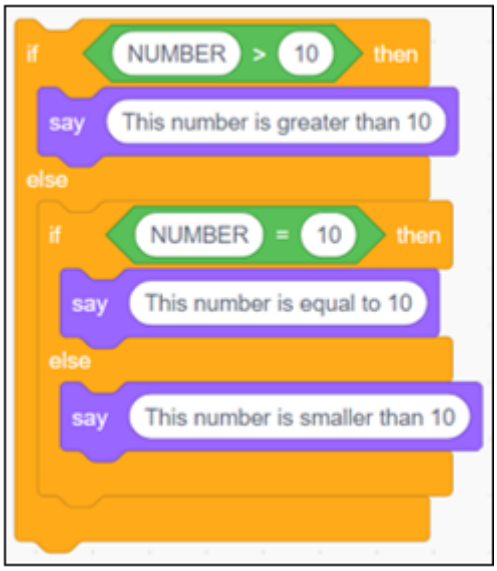



Figure 1 – Example of an algorithm in Scratch.
Source: Made by author.

In the previous image, even having no idea what the project is about, anyone with any or no level of programming knowledge would be able to understand what is going on, and identify basic concepts such as the variable in question ("NUMBER") and the commands acting on it, and thus identifying a the structure at hand (if, else). Scratch's level of ease of use and understanding not only presents a great tool for people of any age to learn basic programming concepts, but also a great stimulus for developing logical thinking for young people who would not only continue to develop such logic, as they are already becoming familiar with concepts such as variables, commands, repetition structures, eventually matrices, queues, stacks, until they reach the most middle/advanced levels of programming, which would lead to the next step in their academic life: high school.

With basic programming concepts in mind, as well as a logical line of reasoning, young people migrating from elementary to middle school would be ready to shoulder more complex concepts as well as peer languages and programming environments. Even though up to the present point, these young people have only interacted with block-shaped commands, programming logic itself is already acknowledged and incorporated by said individuals by the time the real coding comes, that is, writing lines and commands

manually.

The process of learning more complex aspects of programming, as well as basics of other related areas (for example: databases, computer networks, and others) would go on until the end, that is, until high school, at which point, the students have to decide which area they would want to fully grasp and take as their profession, leaving those who decide not to engage on computer related areas with a cohesive and logical mentality, and those who do decide to engage on computing areas with most of the knowledge they will need already on hands, not having to waste their initials years on college learning basic concepts of programming which they were already supposed to have, giving both the students and colleges the possibilities to dive deeper into more complex and fascinating topics without the need of spending months teaching and learning what a vector is.

## 4. Results

Based on the experiments of the sources used during the article, more specifically the introduction of programming in schools in the United Kingdom and the experiment of Thai students, a hypothetical school grade based on the studies is presented, being cataloged by the students' age instead of the period/year they are studying for the sake of both simplicity, consistency with the studies performed and the fact that certain educational institutions may or may not operate differently in relation to the teaching of their students.

### *4.1 Data from 5 to 6 Years Old*

At this stage, the very basics of programming logic are introduced through basic logic exercises and examples of small problems and how these can be solved through logical reasoning, lessons and exercises being treated as small tasks or interactive activities, so the students can grasp said knowledge in the form of little games and fun exercises. In other words, it is taught how to “think in a more logical way”, said logic being exercised and evolved throughout the student's academic life.

### *4.2 Data from 7 to 11 Years Old*

With a foundation of logical reasoning, it is time to introduce basic programming concepts to the individual. Just as in mathematics the student learns multiplication and division in order to be able to solve more advanced equations in the future, this step introduces the most basic programming concepts that are used by almost if not all of the programming languages: IF WHEN, WHEN, WHILE, among other basic concepts, these being initially presented and exercised outside the computational environment, being reviewed and handwritten on paper, and after this brief presentation, we begin to apply this knowledge using the Scratch programming language, where the student will have easy access to all the commands taught and some additional ones, and you can exercise both basic programming concepts and their programming logic, focusing on these two topics without worrying at the moment about lines and lines of traditional compiler code. Eventually, slightly more complex concepts such as arrays, functions, among others are introduced, but still maintaining the simplicity of such concepts so that the difficulty helps to evolve and evolve along with the student's logical reasoning.

### *4.3 Data from 11 to 14 Years Old*

Consolidated computational programming logic and acquired basic programming concepts, the time has come not only to delve into more advanced programming concepts, but also the need to take a step forward and put Scratch aside, preferably using structured programming languages due to the ease of writing and reading most of them (eg C++ or Phyton) where the student will apply all the knowledge gained to this point in the real-time writing of simple programs and algorithms, which, again, will advance in complexity with the student, while still having a somewhat safe zone, where trial and error are key to further develop their logic.

### *4.4 Data from 14 to 16 Years Old*

Finally approaching the final years of pedagogical education and with university life on the horizon, we focus on "preparing the student to prepare for the labor market", that is, giving valuable knowledge to the students who want to work in programming, but again, basic knowledge, so that it does not overwhelm students who choose not to work in the area, just as the college is left to develop such knowledge, in short: instead of teaching everything from scratch, giving extremely and dangerously superficial knowledge to the student, colleges would already have students with basic computer skills, not only restricted programming, but could focus on expanding specific areas (database, information security, advanced programming, among others) without worrying about wasting the early college years teaching things the student should already have in mind when entering the course. With this in mind, in the final step of this academic grid, the student learns how to interact and code with object-oriented programming languages, as well as concepts of interfaces and database linking.

Notice the abstinence of the mention of "use case", "relationship entity" and other concepts which are common and mostly taught in the academic area of programming. As much as these diagrams and documents undoubtedly stimulate and exert the concepts of programming logic and algorithm, they would be best taught in an individual's university life, as these would not only take the time and resources that would be best applied in the teaching of logical concepts, but would also be somewhat of a waste for those who will not follow a professional career in the computational field of work.

## 5. Final Considerations

Based on the studies performed, this hypothetical academic grade mentioned above certainly proposes a relatively simple way of introducing programming as a school subject not unlike other subjects such as mathematics or physics, presenting programming as simple and easy to learn compared to classic stereotypes that programming is just for a certain niche of super-intelligent people who program on black screens and with various symbols and letters almost impossible to understand, showing the potential and opportunities that the computational field of work presents on the modern society.

Overall, not only would this educational path would prepare individuals for computational related college courses, saving them from wasting their early college years by learning the concepts of programming over short periods of time and too superficially to even get used to it, but it would also introduce a more logical and cohesive reasoning for the child that would evolve and improve for the rest of his life, even if the

individual chooses other areas of education than computing.

## 6. Bibliographic References

PRADO, M. E. B. B. LOGO – Linguagem de Programação e as Implicações Pedagógicas. Campinas: UNICAMP, 2000.

OBAMA, B. Don't Just Play on Your Phone, Program It. The White House Blog. 2013. Available on: <http://m.whitehouse.gov/blog/2013/12/09/don-t-just-play-your-phone-program-it> Access on September 10, 2019.

IGNATOVA, M. The Top 10 Skills You Will Be Hiring for in 2017. 2016. Available on: <https://business.linkedin.com/talent-solutions/blog/trends-and-research/2016/the-top-10-skills-you-will-be-hiring-for-in-2017> Access on september 21, 2019

LIN, J. MEI-CHUEN; YEN, LONG-YUEN; YANG, MEI-CHING & CHEN, CHIAO-FUN. Teaching Computer Programming in Elementary Schools: A Pilot Study. Taipei, Taiwan. Taiwan National Normal University. 2004.

DREDGE, S. Coding at school: a parent's guide to England's new computing curriculum. 2014. Available on: <https://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming> Accesso on september 11, 2019.

WOOLF, C. Reading, math and ...Javascript? Coding is now mandatory in English schools. Public Radio International (PRI). 2014. Available on: <https://www.pri.org/stories/2014-09-25/reading-math-and-javascript-coding-now-mandatory-english-schools> Access on september 10, 2019.

MANZANO, J. A., & DE OLIVEIRA, J. F. . Algoritmos - Lógica para Desenvolvimento de Programação de Computadores. 17 ed. São Paulo : Érica Publisher, 2005.

ARMSTRONG, J., WILLIAMS, M., WIKSTROM, C., & VIRDING, R. Concurrent Programming in Erlang. 2 ed. Englewood Cliffs , New Jersey: Prentice-Hall Publisher, 1996.

Code.org Advocacy Coalition. 2018 State of Computer Science Education Policy and Implementation. Available on: <https://computersciencealliance.org/wp-content/uploads/2019/02/2018_state_of_cs.pdf> Acess on november 8, 2019