https://doi.org/10.53656/math2022-1-3-dev

Educational Technologies
Образователни технологии

DEVELOPING PROBLEM SOLVING COMPETENCY USING FUNCTIONAL PROGRAMMING STYLE

Muharem Mollov, PhD student[1)], Petar Petrov, PhD student[2)]
[1)] *"Paisii Hilendarski" University – Plovdiv (Bulgaria)*
[2)] *"Prof. Assen Zlatarov" University – Burgas (Bulgaria)*

Abstract. This paper is dedicated to the challenges of the education that high school students are facing while developing specific competencies related to the functional programming style (FPS). The presented educational approach consists of two components: first, learning FPS by comparing it with the imperative, procedural, object-oriented and logic programming paradigms and second, using competencies based approach for solving practical problems with functional programming. The paper presents a problem set and the phases of its application in the learning process. The results and the analysis of the approach are presented in two groups of high school students which develop successfully their specific competencies for using FPS for practical problem solving. The presented results show that the students are understanding easier FPS and its differences from their known paradigms (imperative, procedural, object-oriented and logical) by using a problem set with properly prepared practical problems which they can solve in multiple ways which lead them to the FPS solution.

Keywords: functional programming; education; software engineering; competency

Introduction

In the Computer Science (CS) education in general and profession "Applied programmer" (PAP) (Staribratov 2020), the students are learning and acquiring the competencies of the functional programming style (FPS). The acquired knowledge, skills and competencies for FPS are listed in the National standards in education (NSE)[1)] for the PAP. Two groups of students were observed. After an analysis of the problems which the students are facing while learning FPS as part of the National programme "Education for IT Career" (NPEITC)[2)] of Ministry of Education and Science (MES) of Bulgaria was found that the students have difficulties with learning and applying FPS because for them this is a whole new way of thinking and constructing a solution of a problem. As part of their education in NPEITC the students have access to e-learning resources on a Moodle-based platform[3)] .This paper provides a problem set and the phases of developing FPS competencies.

The problem set offers practical approach for comparative learning of the functional, imperative, procedural, object-oriented and logic programming paradigms. On the other side, the accomplished education is directed towards development of the competency of the learners which means that they need to solve real and practical challenges. The problems which they receive are not formulated in the terms of FPS. Instead they are practical and require the possession of specific competencies allowing the future professionalists to effectively analyse the problems and pick FPS approach for its solution.

The ideas of competency-based education (CBE) are developed in the last 50 years. The present article considers the European qualification framework (EQF) for lifelong learning[4)]. It is worth to mention the results in this area achieved in the professional education in Bulgaria by applying competency-based education for the profession "System Programmer" in the frame of "Bulgarian-German project for improving the opportunities for employment of young people in Bulgaria" (Staribratov 2009) where they develop the foundations of the National educational standard (NES) for the profession. CBE for PAP is compliant with the related competency European Commission frameworks [4), 5), 6)] and the recommendations of the IT business in Bulgaria[7), 8)] regarding the competencies which the job applicants should have.

NPEITC provides education for high school students based on the developed standards, curriculum and programmes. One of the educational modules in the curriculum is „Functional programming" (FP). According to the Level 4 of EQF there are units of learning outcomes (ULO) in the NES. The described learning outcomes, including knowledge, skills and competencies, are reflected in the curricula for both theoretical and practical education and can be generalized as **the ability to solve practical problems by using FPS**.

Developing FPS competencies during education

In November 2020 a training in module FP was conducted as a part of the curriculum of NPEITC. Two groups were observed: students at High School "Hristo Botev" in Chepintsi, Bulgaria and Vocational High School of Electronics and Engineering "Konstantin Fotinov" Burgas, experienced similar difficulties and obstacles in learning FPS. The participants are facing the challenge to form competencies for using FPS related with its essence and application. The main problem which occurs for the students during the FP module is related to the new programming style which is different compared to the ones (imperative, procedural, object-oriented, logical) which they know and understand to this moment. FPS offers many advantages such as modularity, easier code maintenance, lower code redundancy, helps easier and more effective data processing and more (Hughes 1989). Teodosiev (Teodosiev 2006; 2010; 2011) analyses the problems of the programming education and the different programming paradigms, including FPS and researches

the influence of the programming style. According to his research the programming language and paradigm strongly influences the ability of novice programmers to create algorithms and programs. His suggestion is to not only teach programming languages but also different styles of programming and their corresponding good practices.

There are many functional programming languages or possibility for applying of FP – many libraries for C++ (MCNamara 2004), Language Integrated Query (LINQ) in C#, which provides elegant and effective approach for solving many routine problems in programming related to data-processing, there is also a special FP language in the. NET platform called – F#, which provides accessible syntax and supports concurrency.

During the conducted edition of teaching the FP module it was noticed that the students are naturally looking for a parallel between the functional and the procedure paradigms. This happens while the students are attempting to solve problems. A possible reason is that the students already have knowledge in procedural programming, which leads them to intuitively looking for similarities. Sometimes they do not realise that these are two different paradigms and often think that the FP is just an upgrade of their knowledge. While trying to solve a problem initially they look for solutions closer to their familiar style of programming and they are trying to apply their existing knowledge and skills to construct a solution.

To solve this problems the authors of this paper recommend to point the similarities and the differences between the functional paradigm and the other ones, with which the students are already familiar. Another recommendation is to make parallels between solutions using FP and solutions using the procedural approach.

Based on direct observations during the education for the profession "Applied programmer" in the two groups, it is found that there is a "stereotype" in favour of the procedural paradigm. The students seem to have difficulties to construct solutions based on the FP for problems that require functional approach. This brings the need to look for approaches to form the competencies for using FPS.

There is a variety of approaches for developing competencies in FPS in the CS education. Generally, they can be separated in two groups – comparative approach and practical approach for learning FP.

An example of a comparative approach is the approach of Joosten (Joosten1993) who proposes comparative analysis of imperative and functional style (Joosten 1993). Banchev (Banchev 2017) analyses the effectiveness of the introduction programming courses using different paradigms – imperative and procedural using Java and Ruby. Another applied approach is using FP as an explanation of the object-oriented paradigm (Kristensen 2001). Thompson (Thompson 1997) suggests an approach for problem solving valid for both procedural and functional programming languages consisting of the following steps: understanding, solution design, coding and reflection. Hanus (Hanus 1997) proposes a gen-

eralized model for learning functional and logic programming using the language Curry (Hanus 1995), where the ***logic programming*** is considered as an ***extension of FP*** (Hanus 2007). Todorova (Todorova 2010) and Zinoviev[9] seen functional programming based on λ-calculus as natural subarea of logic programming. Joosten (Joosten 1993) prefers the idea of ***using FP in the beginning of the CS education*** and shows in that the students are highly motivated and develop higher level of algorithmic thinking. Satoshy Egi (Egi 2020) suggests FPS ***oriented towards pattern-matching***, allowing to define not only the most basic data processing functions but more practical mathematical algorithms such as Boolean satisfaction problem (or SAT for short) (Eén, 2006)[10], system of equations and so on. Diaconu suggests ***inductive FP (IFP)*** for generating modular functional programs and function reusage (Diaconu 2020).

Many authors suggest concepts and approaches for learning FP based on Haskell (Hudak 1989), (Hudak 1999), (Davie 1992), (Bird 1998), (Thompson 1993).

Among the practical approaches we can point out the method of ***gamification***, used for the development of apps, such as Soccer Fun (Achten 2008), ***block programming*** (Poole 2019). M. Fansler (Fansler 2015) proposes approach which uses a high-level language to create and manipulate ***multimedia and interactivity*** to teach FP. Promising results are shown by Chattopadhyay (Chattopadhyay et all 2018), who proposed learning functional programming through the STEM and STEAM approach in which the students are having fun, learn by experience by programming ***learning robots*** with the aim to gain knowledge, develop skills and competencies.

In the presented article it is chosen an approach, based on comparative analysis and analogies between FPS, procedure style and logic programming, taking into consideration the short summary of the different educational approaches, the aims, which are listed in the curriculum and the duration of teaching FP module.

Aim of the research

To check the acquired knowledge, skills and competencies integrated in the general competency for FPS of the students regarding their work with practical assignments by the proposed comparative approach with the other programming styles and paradigms.

Research methods

The research uses the following methods: observations, polls with teachers and students, tests, result analysis.

Methodological tools

Methodological tools used in this research:

A) Set of problems which help the development of knowledge, skills and competencies related to the usage of FPS

B) Software and technological means: hardware, educational software for FP

Design (phases) of the research – Research of the efficiency of the proposed methodology.

To gain feedback on the achievements of the students acquiring to the competencies listed in the NES and in the curriculum we developed a problem set, a test, and a practical assignment supported by a sample solution.

It is suggested to follow these steps:

1. Solving the problem following the procedural style.
2. Using recursion in procedural style wherever it is possible (for description of algorithms, complex data structures, etc.).
3. Substitution of the procedural style with declarative style and usage of pure functions and recursion.
4. Suggesting a solution which is close to the logic programming.

To achieve that the following sample problems are provided, showing application of the suggested algorithm for transforming the solution, based on a procedural style, to a solution, based on the FPS. A ***problem set*** is prepared to develop the competencies related to FSP (see unit 10 of the learning outcomes in the NES), in which alternative solutions are provided in order to allow easier adaptation of the students to the functional paradigm. After presenting and analysing different ***sample solutions***, the students receive multiple problems as a ***test***, which require the acquisition of competencies, related to FPS. The last part of the check of the level of knowledge of FPS is an ***assignment*** in which the students should propose practical problems along with the solutions based on FPS.

Research execution

The research consists of 3 main steps: a) Solving problem 1 and 2; b) Test, based on the problems 3 and 4; and c) Verification of the problem-solving competencies by asking students to propose a practical problem and its solution – problems 5 and 6.

The students and their teacher solve problem 1 in 3 ways. The aims are: to show a way to overcome the stereotypes of the procedural way of thinking; to observe the solutions: close to the procedural (imperative) programming style, clean functional programming-based solution, and a solution which corresponds with the logic programming style.

1. Sum of list elements

Write a program which sums the elements of a list

Solution:

https://github.com/msmfenn/FunctionalProgramming/blob/main/Task%201.%20Sum%20Of%20Elements.hs shorturl.at/irDVY

```haskell
sumElements totalSum list =
  if null list
    then totalSum
    else sumElements (totalSum + head list) (tail list)

sumElements1 totalSum [] = totalSum
sumElements1 totalSum (x : xs) = sumElements1 (x + totalSum) xs

sumElements2 = foldl (+) 0

main = do
  input <- getLine
  let list = read input :: [Integer]

  let result = sumElements 0 list
  print result

  let result1 = sumElements1 0 list
  print result1

  let result2 = sumElements2 list
  print result2
```

The task implies multiple solutions, the first of which is close to the procedural style of programming and is easily understandable for the students. The second one is closer to the logic programming and the last one uses the function ***foldl***. According to the poll, the first way is easier and it looks the students are familiar with it. After analysis of the second solution, the students ***find formal connection*** between the two ways which they remember, understand and eventually apply. The third way uses the ***foldl*** function which offers built-in construction making the solution much shorter.

Problem 2 shows how to solve a practical problem with the help of ***foldl***, ***zipWith*** and ***map*** functions. The aim is to gain knowledge about these functions.

2. Receipt

Write a program that calculates the total amount of purchases,including 20% VAT. The input for each purchase is its quantity and the unit price without VAT.

Solution: https://github.com/msmfenn/FunctionalProgramming/blob/main/Task%202.%20Receipt.hs shorturl.at/sxFH5

```haskell
-- receipt
calculateVAT x = x * 1.2

main = do
```

```
  input1 <- getLine
  let quantities = read input1 :: [Float]

  input2 <- getLine
  let prices = read input2 :: [Float]

  let pricesWithVAT = map calculateVAT prices
  let pricesWithVAT1 = map (* 1.2) prices -- alternatively
  let values = zipWith (*) quantities prices
  let totalSum = sum values

  let valuesWithVAT = zipWith (*) quantities pricesWithVAT
  let totalSumWithVAT = sum valuesWithVAT

  print values
  print totalSum
  print (totalSum * 1.2) -- the total sum with the VAT coefficient

  print valuesWithVAT
  print totalSumWithVAT --the total sum with VAT obtained with summation,
  should be the same as the totalSum * 1.2 if everything is correct
```

This task presents different basic techniques for data processing based on FPS with the help of foldl, ***map***, ***zipWith*** and lambda functions.

Problem 3 and 4 are given as a test for individual solving. The students are divided in two groups.

3. Cargo 1

Different by ***size*** and ***count*** full tanks contain liquid, which should be transported by a ***tanker***, which provides a given capacity. Help the team by deciding if it is possible to fit the cargo by taking into account tanker capacity. You will receive ***the capacity (free space as volume measure) of the tanker, a list of the diameters, a list of the lengths, and the amount of each tank type.*** If the cargo does fit in the tanker, output ***"It is possible to carry"***, otherwise print ***"The load is too large"***

Solution: https://github.com/msmfenn/FunctionalProgramming/blob/main/Task%203.%20Cargo%201.hs shorturl.at/euR24

```
isPossibleToCarry x y
  | x <= y = "It is possible to carry"
  | otherwise = "The load is too large"

isPossibleToCarry1 x y =
  if x <= y
    then "It is possible to carry"
    else "The load is too large"

main = do
```

```
11   input <- getLine
12   let tankerVolume = read input :: Float
13
14   input1 <- getLine
15   let diameters = read input1 :: [Float]
16
17   input2 <- getLine
18   let lenghts = read input2 :: [Float]
19
20   input3 <- getLine
21   let tankCounts = read input3 :: [Float]
22
23   let volumes = zipWith (\d l -> pi / 4 * d * d * l) diameters lenghts
24
25   print volumes -- volumes
26   let totalVolumes = zipWith (*) volumes tankCounts
27   let totalVolume = sum totalVolumes
28   print totalVolume
29   print (isPossibleToCarry totalVolume tankerVolume)
30   print (isPossibleToCarry1 totalVolume tankerVolume) -- alternatively
```

4. Cargo 2

A client would like to order different by size and count tanks. Help the team by deciding whether it is possible the order to be fulfilled with the available sheet metal. You will receive ***the area*** of the sheet metal, list of ***diameters*** of the tanks, ***lengths*** and ***quantity*** of the tanks of each kind. If the needed metal sheet is available, print ***„It is possible to be produced”,*** otherwise ***print „There are not enough materials”.***

Solution:

https://github.com/msmfenn/FunctionalProgramming/blob/main/Task%204.%20Cargo%202.hs shorturl.at/mKR35

```
1  isPossibleToMake x y
2    | x <= y = "It is possible to make"
3    | otherwise = "The materials are not enough"
4
5  isPossibleToMake1 x y =
6    if x <= y
7      then "It is possible to make"
8      else "The materials are not enough"
9
10 main = do
11   input <- getLine
12   let materials = read input :: Float
13   input1 <- getLine
14   let diameters = read input1 :: [Float]
15   input2 <- getLine
```

```
16  let lenghts = read input2 :: [Float]
17  input3 <- getLine
18  let tankCounts = read input3 :: [Float]
19  let aroundAreas = zipWith (\d l -> pi * d * l) diameters lenghts
20  print aroundAreas -- around areas
21  let totalAreas = zipWith (*) aroundAreas tankCounts
22  let totalArea = sum totalAreas
23  print totalArea
24  print (isPossibleToMake totalArea materials)
25  print (isPossibleToMake1 totalArea materials) -- alternatively
```

Results

Problem 3 was solved by 80% of the students, where 60% of them used ***isPossibleToCarry1 function,*** 10% of students made mistake in the formula ***zipWith (\r l->2*3.14*r*r*l)***, instead ***zipWith (\d l->pi/4*d*d*l),*** which is explained by the mixing of the radius and diameter and the formulas for calculating circumference and area.

Problem 4 was solved by 60% of the students, 40% used ***isPossibleToCarry1,*** 20% used ***isPossibleToCarry*** and 20% did not use a special function.

Problems 5 and 6 are suggested by the students. Please note that the provided solutions are correct.

5. Terracotta and faience

In a building materials store the seller and the client are facing the following problem: The client likes a few kinds of square terracotta tiles for floor and has to calculate how much packages of each should buy. You will receive a sequence of ***sizes*** of tiles of each type, ***amount of tiles in each package*** of each tile's type and the ***area*** of the room which needs to be covered. Output the ***sequence*** of the number of packages of each tile's type that are needed to cover the area.

Solution: https://github.com/msmfenn/FunctionalProgramming/blob/main/Task%205.%20Terracotta%20and%20faience.hs shorturl.at/exzPX

```
1 squaringList list =
2   if null list
3     then []
4     else head list * head list : squaringList (tail list)
5
6 squaringList1 [] = []
7 squaringList1 (x : xs) = (x * x) : squaringList1 xs
8
9 squaring x = x * x
10
11 squaringList2 = map squaring
12
```

```
squaringList3 = map (\x -> x * x)

main = do
  input1 <- getLine
  let areaToCover = read input1 :: Double
  input2 <- getLine
  let lenghts = read input2 :: [Double]
  input3 <- getLine
  let countsPerPacket = read input3 :: [Double]

  let pieceArea = squaringList lenghts
  let areasPerPacket = zipWith (*) pieceArea countsPerPacket
  let rec = map recip areasPerPacket
  let packets = map (areaToCover *) rec
  let roundedPackets = map ceiling packets
  print roundedPackets
```

It is interesting that the problem is solved by similar methods as problem 1 – method close to the imperative programming, logic programming, and to the functional style-based method.

6. Analog watch

Calculate the angle between the arrows of a clock given the coordinates of the hour's arrow and the minute's arrow. The origin of the coordinate system is in the centre of the clock-face.

Solution: https://github.com/msmfenn/FunctionalProgramming/blob/main/Task%206.%20Analog%20watch.hs shorturl.at/ixNQ2

```
main = do
  let a = [0.707, 0.707] --sqrt(2)/2 45 degree
  let b = [-2, 4 * sqrt 3 / 2] -- 120 degree
  let abScalar = sum (zipWith (*) a b)

  let aLen = sqrt (sum (zipWith (*) a a))
  -- let aLen = sqrt( foldl (+) 0 (zipWith(*) a a) )-- alternatively
  let bLen = sqrt (sum (zipWith (*) b b))
  -- let aLen = sqrt( foldl (+) 0 (zipWith(*) b b) )-- alternatively
  let cosAngle = abScalar / (aLen * bLen)

  let angle = round (acos cosAngle / pi * 180)
  print angle
```

In this problem the students took advantage of the list processing in FPS. The solution is short and shows the power of FPS in applying mathematical formulas.

Discussion

The research shows that the usage of comparative analysis between the different styles of programming leads to mastering the programming paradigms and develops

higher level of algorithmical thinking in the students. The proposed approach takes into account the curriculum – both as duration in academic hours and as a sequence of topics. If the sequence of topics is changed, the comparative analysis may be applied a different order.

Regardless of the order, the comparative learning of the paradigms helps to achieve higher level of competency. The learning of the different paradigms in a connected way provides added value in which the Whole is superior to the sum of the Parts.

Of course, if they are more academic hours for learning the FPS module, the practical approach should not be ignored. Grounds for such hypothesis are found in the achievements shown by the students in the "Embedded systems" and the "Operating systems" module (Mollov 2020) where similar approaches were used.

Conclusion

The research results lead to the following conclusions:

1. The procedural style of programming becomes primary way of thinking and the students have hard time with getting to think in terms of FPS.
2. The proposed approach helps to overcome these peculiarities. The result is that students use different approaches for solving FPS-based problems.
3. By the proposed approach the students develop effectively their competencies, related to FPS.

According to the requirements of Unit №10 of the learning outcomes part of NES, the solutions of the problems provided by the students show that they have the needed knowledge, skills and are capable to offer optimal solutions based on FPS. 70% of the students who solved the practical problems have chosen a proper data structure and used proper functions in Haskell.

The students passed successfully their exams on the FP module.

FPS a challenge for the students. They feel more comfortable with the procedural style and it is needed to use different methodologies to ease the students. It could be concluded that: *The suggested approach helps to develop FPS, offers opportunity to compare the advantages and the disadvantages of the different paradigms, and develops the ability to choose the proper tools for solving different problems related to software engineering.*

Acknowledgement. The authors thank Dr. Ivaylo Staribratov, Assoc. Prof. from Plovdiv University (Bulgaria) for the methodological help regarding the creation of this paper and the research conduction.

NOTES

1. ORDINANCE № 1 of 15.01.2018 for acquiring a qualification in the profession "Applied Programmer" https://www.mon.bg/upload/14210/dos_481030.pdf [Last Visited, 05.08.2020].
2. Ministry of Education and Science, National program "IT Career Training" Portal for e-Training in the specialty "Applied Programmer", https://www.mon.bg/upload/19218/19RH172pr6-IT-kariera.pdf [Last Visited, 05.08.2020].
3. National program "IT Career Training" Portal for e-Training in the specialty "Applied Programmer" https://it-kariera.mon.bg/e-learning/ [Last Visited, 05.08.2020].
4. European e-Competence Framework, https://www.ecompetences.eu/e-cf-overview Publisher: Publications Office of the European Union ISBN: 978-92-79-68006-9 (pdf),978-92-79-68005-2 (print),978-92-79-74173-9 (ePub) ISSN: 1831-9424 (online),1018-5593 (print) DOI: 10.2760/38842 (online) 10.2760/836968 (print) 10.2760/00963 (ePub) [Last Visited, 05.08.2020].
5. The European Qualifications Framework for Lifelong Learning (EQF), http://relaunch.ecompetences.eu/wp-content/uploads/2013/11/EQF_broch_2008_en.pdf.
6. The digital competence framework for citizens with eight proficiency levels and examples of use (DigComp 2.1), https://op.europa.eu/bg/publication-detail/-/publication/3c5e7879-308f-11e7-9412-01aa75ed71a1/language-en [Last Visited, 05.08.2020].
7. National competence assessment system MyCompetence https://mycompetence.bg Last Visited, 05.08.2020].
8. Strategic requirements of the software industry for education system reform https://www.basscom.org/RapidASPEditor/MyUploadDocs/Software-Industry-Requirements-for-Educational-Ref.pdf [Last Visited, 05.08.2020].
9. Zinoviev, Anton. 2018. Logichesko programirane, http://logic.fmi.uni-sofia.bg/zinoviev/lp/lp-20190713.pdf [Last visited 05.08.2020].
10. Philosophæ doctor thesis Hoessen Benoît, Solving the Boolean satisfiability problem using the parallel paradigm, http://www.theses.fr/2014ARTO0406/document [Last visited 30.08.2020].

REFERENCES

ACHTEN, P., 2008. *Teaching functional programming with soccer-fun*, Soccer-FunArticle, Available from: doi: 10.1145/1411260.1411270.

BANCHEV, B., 2017. Rolyata na ezika v uvodnoto obuchenie po programirane, *X Natsionalna konferentsia „Obrazovanieto i izsledvaniyata v informatsionnoto obshtestvo"*, **21** [In Bulgarian].

BIRD, R., 1998. *Introduction to Functional Programming using Haskell. Prentice Hall*, New York.

CHATTOPADHYAY, A., QUIGLEY, E., HART, R. & PETTY, S., 2018. *A BERO CLF Themed Nifty Middle School Module: Teach Functional Programming Using Music and Generate Interest in Coding and Robotics,* 98 – 103. Available from: doi. 10.1145/3241815.3241861.

DAVIE, A., 1992. *Introduction to Functional Programming System Using Haskell*. Cambridge University Press.

DIACONU, A., 2020. Learning functional programs with function invention and reuse Teaching Functional Programming with Soccer-Fun Peter Achten arXiv: 2011.08881 v1 [cs.PL] 17 Nov 2020 Honour School of Computer Science Trinity.

EGI, S. & NISHIWAKI, Y., 2020. Functional Programming in Pattern-Match-Oriented Programming Style, *The Art, Science, and Engineering of Programming*, Published February 17, 2020, Available from: doi: 10.22152/programming-journal.org/2020/4/7 © The Art, Science, **4**(3), 2020, article 7; 32.

EÉN, N. & SÖRENSSON, N., 2006. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, **2**(1 – 4), 1 – 26.

FANSLER, M., 2015. *A Multimedia Library for Teaching Functional Programming Concepts*, Available from: doi. 10.13140/RG.2.1.4000.7529.

HANUS, M., 1992, 1997. Teaching Functional and Logic Programming with a Single Computation Model*, Proc. Ninth International Symposium on Programming Languages, Implementations, Logics, and Programs* (PLILP'97), Southampton, UK. Springer LNCS, 335–350.

HANUS M., HERBERT, K. & MORENO-NAVARRO, J., 1995. Curry: A Truly Functional Logic Language. *Proceedings of ILPS'95 Workshop on Visions for the Future of Logic Programmming*, Portland, Oregon, United States, 95 – 107.

HANUS, M., 2007. Multi-paradigm Declarative Languages. *Proceedings of the 23rd International Conference on Logic Programming (ICLP 2007)*, Porto, Portugal. Edited by Verónica Dahl and Ilkka Niemelä. **4670**. LNCS. Springer, 2007, 45–75. ISBN: 978-3-540-74610-2. Available from: doi. 10.1007/978-3-540-74610-2_5.

HUDAK P., 1989. Conception, evolution, and application of functional programming languages. *ACM Computing Surveys*, **21**(3): 359 – 411.

HUDAK P., PETERSON, J. & FASEL, J., 1999. *A Gentle Introduction to Haskell 98*.

HUGHES, J., 1989. Why Functional Programming Matters, *Computer Journal*, **32**(2), 98 – 107. https://www.cs.kent.ac.uk/people/staff/dat/miranda/whyfp90.pdf [Last Visited, 30.5.2021].

JOOSTEN, S., VAN DEN BERG, K. & VAN DER HOEVEN, G., 1993. *Teaching Functional Programming to First-Year Students Article in Journal of Functional Programming,* Available from: doi. 10.1017/S0956796800000599 Source: CiteSeer.

KRISTENSEN, J.T. & HANSEN, M., 2001. Teaching object-oriented programming on top of functional programming, *Proceedings – Frontiers in Education Conference* 1: TID – 15 – 20, **1**, Available from: doi. 10.1109/FIE.2001.963848.

MCNAMARA, B. & SMARAGDAKIS, Y., 2004. *Functional Programming with the FC++ Library. Journal of Functional Programming*. **14**, 429–472. Available from: doi. 10.1017/S0956796803004969.

MOLLOV, M. & STOITSOV, G., 2020. Development of STEM Competencies to the Profession "Applied Programmer" in a Virtual Environment, *Anniversary International Scientific Conference "Synergetics and Reflection in Mathematics Education"*, 16 – 18 October 2020, Pamporovo, Bulgaria, 285–292, Plovdiv: University press, ISBN: 978-619-202-595-3.

POOLE, M., 2019. *A Block Design for Introductory Functional Programming in Haskell*. 31-35. 10.1109/BB48857.2019.8941214.

STARIBRATOV, I., 2020. Alternativen nachin za profesionalno obrazovanie. Vocational education 22.2: 173 – 178.

TEODOSIEV, T.K., 2006. Problemi na obuchenieto po programirane. *Sbornik dokladi na Natsionalnata konferentsiya "Obrazovanieto v informatsionnoto obshtestvo"*, Plovdiv, oktomvri, 2006 g., URI: http://hdl.handle.net/10525/1494, ISBN: 10:954-8986-22-1, 13:978-954-8986-22-9 [In Bulgarian].

TEODOSIEV, T.K., 2010. Stilat na programata kato sredstvo za izbyagvane na greshki. *Sbornik dokladi na Natsionalna konferentsia „ Obrazovanieto v informatsionnoto obshtestvo"*, Plovdiv, ARIO, 27-28 may 2010, 87-94, URI: http://hdl.handle.net/10525/1383, ISSN: 1314-0752 [In Bulgarian].

TEODOSIEV, T.K., 2011. Stil na programirane v obuchenieto, *Sbornik dokladi na Natsionalna konferentsiya "Obrazovanieto v informatsionnoto obshtestvo"*, Plovdiv, ARIO, 26-27 may 2011, 073-079, URI: http://hdl.handle.net/10525/1524, ISSN: 1314-0752 [In Bulgarian].

TODOROVA, M., 2010, *Ezitsi za funktsionalno i logichesko programirane*. Sofia: Siela, ISBN: 978-954-28-0828-2 [In Bulgarian].

THOMPSON, S., 1997. *A Problem Solving Approach in Teaching Functional Programming*.

THOMPSON S., 1993. Formulating Haskell. Launchbury J., Sansom P. (eds), *Functional Programming*, Glasgow 1992. Workshops in Computing. Springer, London. https://doi.org/10.1007/978-1-4471-3215-8_23.

✉ **Muharem Mollov, PhD student**
WoS 39991269 (Author Record Id)
ORCID ID: 0000-0003-0171-4462
https://www.researchgate.net/profile/Muharem-Mollov
Faculty of Mathematics and informatics
"Paisii Hilendarski" University of Plovdiv
236, Bulgaria Blvd.
Plovdiv, Bulgaria
E-mail: muharem.mollov@uni-plovdiv.bg

✉ **Petar Petrov, PhD student**
ORCID ID: 0000-0002-8344-1578
"Prof. Assen Zlatarov" University
1, Prof. Yakim Yakimov Blvd.
Burgas, Bulgaria
E-mail: peshopbs2@gmail.com