

---

# PALMA junior – programming competition in Imagine

**Ján Guniš**, *jan.gunis@upjs.sk*

Dept of Informatics, P. J. Šafárik University in Košice, SLOVAKIA

**Ľubomír Šnajder**, *lubomir.snajder@upjs.sk*

Dept of Informatics, P. J. Šafárik University in Košice, SLOVAKIA

**Valentína Gunišová**, *valentina.gunisova@upjs.sk*

Centre for Lifelong Learning, P. J. Šafárik University in Košice, SLOVAKIA

**Zuzana Szabóová**, *zuzana.szaboova@upjs.sk*

Dept of Informatics, P. J. Šafárik University in Košice, SLOVAKIA

**Andrea Hricová**, *a.hricova@gmail.com*

Dept of Digital Competencies, Prešov University in Prešov, SLOVAKIA

## Abstract

In 2005 we have started to organize new on-line programming competition called PALMA junior using programming environment Imagine based on programming language Logo. The competition is assigned for pupils aged 10 to 15. It runs in two categories: PROFÍK (pupils aged 10 to 11) and EXPERT (pupils aged 12 to 15). The main aims of the competition are encouraging youngsters to solve interesting algorithmic problems, to improve their programming skills and mathematical thinking. Problems, which the competitors solve, are oriented to Programming, Algorithms and Mathematics (PALMA).

One year of competition composes of four bouts. Each bout consists of six problems. Problems from 1 to 4 are for teams in PROFÍK category, problems from 3 to 6 are for teams in EXPERT category. Students of PROFÍK category can solve the problems in EXPERT category, but it is not possible vice versa. Before and during the competition bouts students can study and use resources we have published on website of this competition. Students work on their own or in pairs.

We have created the authorized area in LMS Moodle where students have an access to discussion forums, assignments of problems for particular bouts, chats and questionnaires. The competitors also deliver their solutions of problems in limited time and they can find there the evaluation of their solutions with comments written by organizers after current bout.

In the paper we describe objectives of the competition in the field of improving competitors' programming skills, algorithmic and mathematical thinking and we also show how we cover the objectives by selected collection of problems. In the frame of the competition we would like also to develop knowledge in other areas (biology, geography, history and sports), aesthetics, creativity, pleasure feeling, EU citizenship, exactness etc. For better illustration of competitions problem types, their formulation, authorial solution, competitors' train of thought and their solution, typical misconceptions etc. we analyse and describe pupils' solutions of 6 selected problems (Kingdom of letters, Treasure Island, Pyramid, A drone bee and its (n x grand) parents, Snow-flake, Flags of EU countries). In future we intend to improve learning materials, evaluation tools for whole competition progress and also prepare on-line course for teachers focused on methodology of programming.

## Keywords

Programming competition, primary school, Logo, Imagine, problem analysis

## Background of the competition PALMA junior

Programming, algorithms, mathematics - these three words became fundamental elements of online programming competition PALMA organized by P. J. Šafárik University in Košice, Faculty of Science and civic association STROM (Palma, 2007). This competition is dedicated to secondary school students. They have solved the problems using programming languages C/C++ or Pascal. The goals of PALMA are to give opportunity for competitors to compete, to test their ability to analyze problems, to explore algorithms solving these problems and not least to pre-set them quick, effectively and properly. A younger sibling of PALMA – PALMA junior – was established in 2005, too. Essential principles of PALMA junior resulted from our longtime experience in teaching algorithm development and programming. Valuable experience grew from our meetings with computer science teachers which we have organized during several years (Club of Computer Science Teachers). Among other things we have dealt with methods of teaching algorithm development and programming, choosing suitable programming environments and languages, preparing assignments attractive for pupils and supporting their creativity.

Before preparation of our new programming competition we had studied actual programming competitions in Slovakia. All these competitions have similarities – their main goal is to develop algorithmic thinking and programming skills of competitors.

Usually, the long-term competitions are focused on programming languages C/C++ and Pascal (e. g. National Olympiad in Informatics, KSP, ZENIT). They are oriented to secondary school students and are usually organized in attendance way.

Unlike others the Internet Problem Solving Contest (IPSC) is an online competition. It is not focused on particular programming language. It is up to the competitors to select tools which are appropriate for problem solving. There are other differences yet. The competitors can compete in teams (most of trio) and they solve of competitive assignments which are published on IPSC website.

The competitions focused on younger pupils are usually organized in attendance way. They are making use of children programming environments attractive for this age group – Comenius Logo, Imagine, Baltík.

The first and at that time the only competition for pupils aged 10 to 15 – Cologobežka and Imagine Cup celebrates the 10<sup>th</sup> birthday this year. The goal of the competition is to motivate the pupils which use Logo more seriously for programming activities (Tomcsányiová, Tomcsányi, 2005).

The idea of effectiveness is interestedly integrated into assignment of the competitive teleproject “20 commands” which took place this school year. The goal of the competitive teleproject was to motivate the pupils to work with basic turtle graphics in Imagine environment, to encourage their creativity, imagination and competitiveness.

We take up with the similarly competitions in foreign countries, like National Logo competition-Olympiad in Lithuania (Slapkauskiene, 2005), Logo competition for primary school children – the “miniLOGIA” (Jochemczyk, Oledzka, 2005) or the competition “Beaver” (Dagiene, 2005).

Considering this range of competitions we have tried to come up with another one that should be meaningful and not useless. We have tried to give PALMA junior the most interesting and the most attractive elements for pupils aged 10 to 15 as christening present: creativity, online form, team play, programming environment Imagine, available educational resources and authorial solutions. One of our aims is development of algorithmic thinking of the competitors therefore we consider effectiveness of solutions.

## Description of the competition PALMA junior

PALMA junior is programming competition in programming environment Imagine based on programming language Logo. The competition is assigned for pupils aged 10 to 15. It runs in two categories: PROFÍK (pupils aged 10 to 11) and EXPERT (pupils aged 12 to 15). Problems, which the competitors solve, are oriented to programming, algorithms and mathematics.

One year of competition composes of four bouts. Each bout consists of six problems. Problems from 1 to 4 are for teams in PROFÍK category, problems from 3 to 6 are for teams in EXPERT category. Students of PROFÍK category can solve the problems in EXPERT category, but it is not possible vice versa. Before and during the competition bouts students can study and use resources we have published on website of this competition. Students work on their own or in pairs.

### E-learning support of contest PALMA junior

We have created the authorized area in LMS Moodle. We have chosen this system because it provides active and dynamic environment for communication with participants and it's open, free and developed continually. Thereby the competition acquires the next dimension and becomes very attractive for competitors.

In the authorized area students have an access to discussion forums, assignments of problems for particular bouts, chats and questionnaires. The competitors also deliver their solutions of problems in limited time and they can find there the evaluation of their solutions with comments written by organizers after current bout.

There is a discussion forum and a chat for each bout. It enables the competitors to discuss uncertainties and troubles before the contest bout, they can communicate in chat between each other or discuss with competition organizers about mistiness consequential from assignments, about delivering the solutions etc.

During the bout there are published assignments of problems and all necessary files in the LMS Moodle environment and on the competition website too. The competitors have three hours for solving the problems. During this period the competitors can deliver their solutions in the LMS Moodle environment. After expiration of the designated time we correct and mark delivered solutions and the evaluation is published. The competitors have appreciated that their solutions are assessed (corrected, marked) straight after finishing of the bout, usually up to two hours.

Apart from the assessment of every delivered solution we have published authorial solutions of problems, cumulative evaluation of contestants' solutions with a list of typical mistakes, gallery of final pictures (results of programs) and ranking of competitors on the competition website. Students who do not take part in the competition have an access not only to additional resources. They can access the assignments of problems and the authorial comments of solutions too. It can help them to learn programming in Imagine environment.

### Realization of contest PALMA junior

In school year 2006/2007 we have organized 2<sup>nd</sup> run of the competition. After realization of on-line bouts we have organized the final, attendance bout of programming competition PALMA junior. This final bout was organized at P. J. Šafárik University in Košice, Faculty of Science and the best teams have taken part in this bout. Besides the competition we have prepared funny form of some activities for the competitors – they can solve several logical problems. Absolute winners of the first year were awarded with nice presents.

We are interested in opinion of teachers about this type of competition, therefore during the final bout we discussed with teachers. We talked about an acquisition of this competition for students and teachers, about abilities to improve it. We are interested in teaching of informatics (computer science), especially of algorithm development and programming.

The present result of discussion is that this competition stimulates students from participating schools to be interested more in programming. Natural emulation of students (in a school) helps them to achieve better educational results. The teachers have appreciated accessibility of educational resources and the competition by itself as very good way of students' motivation to improve their algorithmic thinking and programming skills.

According to information received from teachers, there is very active and exciting atmosphere at schools, and contestants are enjoying during the bouts. Some teachers communicate with us via e-mail or chat regarding problems related to organization of actual bout. After the finish of a bout the teachers can express their attitude to choice and quality of solved problems and to atmosphere in classrooms during the bout, some of them send us photos from the bout.

## Objectives and selection of problems for the competition

### Objectives of PALMA junior competition

The main aims of the competition are encouraging youngsters to solve interesting algorithmic problems, to improve their programming skills and mathematical thinking.

Programming skills of competitors can be enhanced by programming of problems whose solutions include loops, conditions, own procedures, operations, variables, procedures and operations of turtle graphics and Cartesian co-ordinate system, visual components, events, using several turtles, lists, recursion etc.

Competitors can develop their algorithmic thinking by using algorithms concerning drawing planar objects with given properties (turtle as a processor of algorithms), sorting and searching objects (numbers), traversing through 2D array (e.g. labyrinth), creation and evaluation of own formulas.

Mathematics thinking can be developed by solving problems which need basic knowledge and skills in the areas of relative orientation in turtle geometry, orientation in Cartesian co-ordinate system, properties of planar objects, calculation of angles, sequences of numbers, arithmetical functions, Pythagorean theorem, calculating position in 2D array, factoring numbers, unit conversions, fractions etc.

In the frame of the competition we would like also to develop knowledge in other areas (biology, geography, history and sports), aesthetics, creativity, pleasure feeling, EU citizenship, exactness.

### Coverage of the objectives by selected collection of problems

In preparation of problem assignments and solutions we have tried to cover all above described dimensions. We have created problems oriented to drawing objects and/or calculations with aesthetics experience (feeling), using knowledge in various areas. Some problems are games (Racing ball, Guess number game, Coins, Clown's houses, Treasure Island), which competitor can use also after competitions. After each bout we publish gallery of pictures which are results of drawing problems.

For better overview (illustration) how we try to cover all objectives by competition problems we have prepared table (Fig.1) that is based on authorial solutions of competitions problems. We have gradually filled in this table and use it as an aid for creation of new problems during the first two years of the competition. We consider it to be also a good cornerstone (starting point) for more precise creation of competition problems in the 3<sup>rd</sup> year of the competition.

Problem name	Programming											Algorithms							Mathematics																
	turtle graphics	Cartesian co-ordinate system	procedures without parameters	procedures with parameters	loops - given number times	loops - conditions	nested loops	conditions	variables - using, changing, operations	randomness	visual components, events	using several turtles	recursion	lists	turtle as a processor of algorithms	traversing through 2D array	labyrinth traversing, testing conditions	randomness in bounds	drawing shapes with some properties	sorting algorithms	searching algorithms	creation and evaluation of expressions	relative orientation in turtle geometry	orientation in Cartesian co-ordinate system	properties of planar objects	calculation of angles	sequences of numbers	arithmetical functions	Pythagorean theorem	calculating position in 2D array	factoring numbers	unit conversions			
House	*				*									*								*		*	*										
Olympic rings	*													*									*		*	*									
Sun	*				*									*											*										
Chessboard	*	*			*		*							*	*								*			*					*				
EC flag	*	*			*		*		*					*								*	*	*	*	*	*						*		
Clocks	*		*					*						*								*	*	*	*	*	*						*		
Santa Claus	*	*			*									*								*	*	*	*	*	*								
Racing ball	*					*		*						*	*	*						*	*	*	*	*	*								
Spiral	*					*		*	*					*								*	*	*	*	*	*								
IQ test		*						*			*										*	*	*	*	*	*	*								
Screen saver	*				*			*	*	*				*	*	*					*	*	*	*	*	*	*	*							
Wallpapering	*				*	*	*	*	*	*				*	*	*					*	*	*	*	*	*	*	*			*				
Snowman	*				*	*	*	*	*	*				*	*	*					*	*	*	*	*	*	*	*			*				
Star	*			*	*	*	*	*	*	*				*	*	*	*				*	*	*	*	*	*	*	*							
Pyramid	*	*	*	*	*	*	*	*	*	*				*	*	*	*				*	*	*	*	*	*	*	*		*					
Guess number game					*		*	*	*	*						*			*	*	*	*	*	*	*	*	*	*							
Coins		*	*					*	*	*	*	*		*	*	*			*	*	*	*	*	*	*	*	*	*							
Flake	*			*	*	*	*	*	*	*	*	*	*	*	*	*	*			*	*	*	*	*	*	*	*	*							
6 EC flags	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*						
Guess state game	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Travelling over EC	*		*				*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Pumpkin	*			*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Winner stages I.	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Target	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Clown's houses	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Winner stages II.	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Aquarium		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Letter figure	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Castle	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Treasure island	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Parents of bee		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Lady-bug		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Shell	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Figure 1. Coverage of the objectives by selected collection of problems

## Analysis of pupils' solutions of selected problems

Our aim is not to create robust and complex solutions of problems. We would rather show the beauty of programming on simply, elegant and effective solutions.

We selected some few assignments to illustrate the type of competition problems, their formulation, authorial solution, competitors' train of thought and their solution, typical misconceptions etc. The goal of assignment formulation is to teach the pupils to identify the problem hidden in the assignment. This competence – to make sense of assignment – is very important. It is presumption for successful problem solving.

### Kingdom of letters

The first assignment of each bout is traditional “drawing”. It provides wide space to the competitors to apply own fantasy and creativity. At first sight a simple problem, but indeed it

offers ability to experienced competitor to use his knowledge and solve the problem by implementation of parameters.

The competitors should have to create arbitrary figure by using only letters and other characters.

The solution of this problem is oriented to working with properties of text, with position and orientation of turtle (turtle co-ordinates and heading). To fit the turtle in place the competitors were in need of correct computation or approximation of turtle position.

Procedure *type\_char* has got six parameters defining a position of turtle (*x*, *y*), its orientation (*direction*), size (*size*) and colour (*colour*) of text and character (*char*):

```
to type_char :x :y :direction :size :colour :char
  setPos list :x :y
  setHeading :direction
  setFont (list "Arial (list :size 10 0 0 1 238))
  setPenColour :colour
  label :char
end

to figure
  hideTurtle
  clearScreen
  ;FACE
  type_char -40 125 0 100 5 "O
  type_char 30 178 -90 50 9 ":"
  type_char 5 165 0 20 5 "L
  type_char 22 150 -90 30 4 "("
  ;BODY
  type_char -54 10 0 120 3 "M
  ;LEGS
  type_char 65 10 180 120 8 "V
end
```

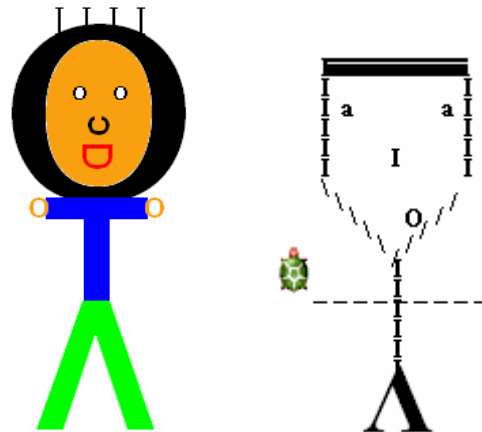


Figure 2. From competitors' gallery

In a few solutions the competitors drew figures by drawing their own characters. They did not take advantage of turtle ability to print text in its current position.

The competitors created several procedures for each part of a figure (head, body, legs, etc.), they worked with properties of text, with position and orientation of turtle, but they did not use parameters. In spite of it, the competitors created patchworky figures.

### Treasure Island

There is usually one assignment in every bout which offers prepared environment to the competitors. The competitors' goal is to integrate their own necessary procedures to activate this environment following the instructions of assignment. We try to show programming as funny activity too. The competitors can easily create simple games and enjoy them.

The competitors should have to create an algorithm processor – pirate Kubo – to find the way through the forest. We prepared environment – pirate map with special marks. Pirate Kubo followed these marks telling him which way to go. To make it harder for Kubo, shape of map was changed after every game (but always solvable).

This assignment required good orientation in two-dimensional array. We were interested in effective evaluation of conditions and computing exact position of pirate.

Direction of every move is assigned by direction of arrows on the map. The arrows defining the same direction are drawn in the same colour. So, for pirate Kubo it is enough to test the colour of point he is standing on. Then he can decide which direction he should go on (turtle changes its position). As he finds he stands on white point – he came over the forest and he achieved his aim, he is near the treasure. A procedure *search* tests colour of point below pirate Kubo and according to this colour it changes his position:



Figure 3. Map of Island of treasure

```

to search
  forward 40
  while [not (dotcolour="white)]
    [if dotcolour="blue [ setxcor xcor+40 ]
      if dotcolour="yellow [ setycor ycor+40 ]
        if dotcolour="green [ setycor ycor-40 ]
          if dotcolour="red [ setxcor xcor-40 ]
        ]
    ]
  end

```

We can notice that application of nested commands ifelse does not lead to more effective solution. Pirate Kubo can execute more steps in one cycle. Majority of competitive teams used this type of solution. But we do not know it is because they reflected this fact, or it is because they simply did not think about ifelse commands. It is interesting that almost a quarter of the competitors used technique of recursion. It means again insufficient comprehension of meaning and applicability of using recursion.

### Pyramid

This assignment represents more complex version of assignment about drawing a chess-board. In contrast to chess-board (where every row consists of same number of squares), row of pyramid contains two triangles less than row below it.

The competitors should have to create procedure for drawing the pyramid. Input of procedure was number of pyramid floors ( $n$ ).

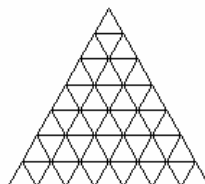


Figure 4. Pyramid with seven floors

It is necessary to put sufficient time to analyze the way of drawing the pyramid. The competitors solved this problem in two different ways.

In the first concept, one cycle draws one row of pyramid. This cycle is repeated in superior cycle, of course, always with declining numbers of repeating.

```
to pyramid :n
  repeat :n [right 90
    forward (:n-counter+1)*20
    left 120
    repeat (:n-counter+1)
      [forward 20 left 120 forward 20 right 120]
    right 60 forward 20 left 30]
end
```

In the second concept, the competitors offered recursive version of this algorithm. Procedure draws one row of pyramid, changes a turtle position and calls its next instance with reduced parameter – the number of floors.

```
to pyramid :n
  if :n > 0 [right 90 forward :n*20 left 120
    repeat :n [forward 20 left 120 forward 20 right 120]
    right 60 forward 20 left 30
    pyramid :n-1]
end
```

The question is whether using recursion is appropriate in this situation. It leads to correct solution, but it is apparently fruitless. A fact is that the competitors know this way of solving problems. In the meantime it shows that they do not appreciate disadvantages, which recursion brings.

### A drone bee and its (n x grand) parents

Completing the terms of sequence is fancy mathematical mind-bender. Well-known sequence is Fibonacci sequence, and the most frequent problem, which leads to Fibonacci numbers, is problem of Fibonacci's rabbits. We chose a less known occurrence (Knott, 2007).

The competitors should have to compute how many (n x grand) parents drone bee has got. We showed one important fact – every drone bee has got only one parent (queen bee) and every queen bee has got two parents (queen bee and drone bee).

We were interested in the way which the competitors used to solve this problem without knowledge in field of sequence (concept, scheme). We knowingly mentioned only number of drone bee's parent (1) and grandparents (2) in problem assignment. Our tendency was to illustrate a difference between family trees of man and drone bee. To verify a right comprehension of text and correct analysis of the problem we let the competitors to discover a dependency and consecutively to compute the next values.

To compute the number of drone bee's (n x grand)parents we availed except the variables *ancestor1* and *ancestor2* (which contained two actual members of sequence) also additional variable *a* (to realize the computation of next terms).

```
to ancestry :n
  ifelse :n=0 [type "1]
    [ifelse :n=1
      [type "2]
      [let "ancestor1 1
        let "ancestor2 2
        repeat :n-1 [let "a (:ancestor1 + :ancestor2)
          let "ancestor1 :ancestor2
          let "ancestor2 :a]
        type :ancestor2]]
end
```

We mentioned a solution without additional variable in author's commentary to this assignment. We can monitor how the competitors will use it in future assignments.

The competitors successfully discovered a mathematical model of the problem. In a few of their solutions they did not pay sufficient attention to testing marginal values. The result of it was incorrect output for input value 0 (number of parents). It is very frequent mistake of novice programmers.

We found one atypical – however not more effective – solution. The competitor divided the problem to two branches according to value of  $n$ .

```
to ancestry :n
  ifelse mod :n 2 = 1 [make "anc1 1
                      make "anc2 2
                      repeat (:n-1)/2 [make "anc1 sum :anc1 :anc2
                                           make "anc2 sum :anc1 :anc2]]
                        [make "anc1 1
                          make "anc2 1
                          repeat :n/2 [make "anc1 sum :anc1 :anc2
                                             make "anc2 sum :anc1 :anc2]]
  type :anc2
end
```

### Snow-flake

By this assignment we wanted to find out how the competitors understand the concept of recursion. We were interested in how they managed computing of angles closed by arms of flake, how they managed orientation by turtle geometry. Recursion is rather challenging programming technique and it calls for a high level of abstraction. If the competitors know ways of drawing typical recursive graphics (binary tree, Koch flake), will they be able to generalize this process and apply it on other graphics?

The competitors should have to draw recursive graphics – snow-flake. There is smaller, half-size snow-flake in every vertex of snow-flake.

We can define this graphics by triplet of parameters: number of arms (*side*), length of arm (*length*) and depth of recursion (*depth*). Then, we can write a procedure flake using technique of recursion:

```
to flake :side :length :depth
  if :depth > 0
    [repeat :side
      [forward :length
        flake :side :length/2 :depth-1
        back :length
        left 360/:side]]
  end
end
```

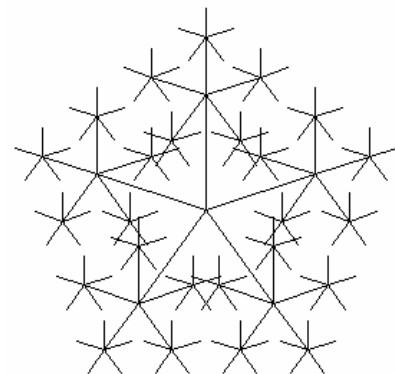


Figure 5. Snow-flake (flake 5 80 3)

Although this assignment contained several samples of snow-flakes with different depth of recursion, we think of this problem as difficult. Almost all competitors mastered the first level of snow-flake. But in next levels the competitors made typical mistake. This mistake is consequence of insufficient analysis of problem and unrecognized recursive character of graphics. Majority of competitors solved the second level by using next, nested cycle. Only one competitive team solved this problem correctly.

## Flags of EU countries

This assignment was solved in the final bout by the winners of school bouts. We used the fact, that Slovakia became a member of European Union, so we reminded the history of formation of European Union.

The competitors should have to create procedure drawing flags of foundation states of EU.

The flags of these states are very similar. Every one of them consists of three varicoloured bars. They differ by proportion of sides, colours of bands and their orientation. As we find important to create algorithms solving a class of problems, not just one special situation, we were interested in level of parameterization achieved by competitors.

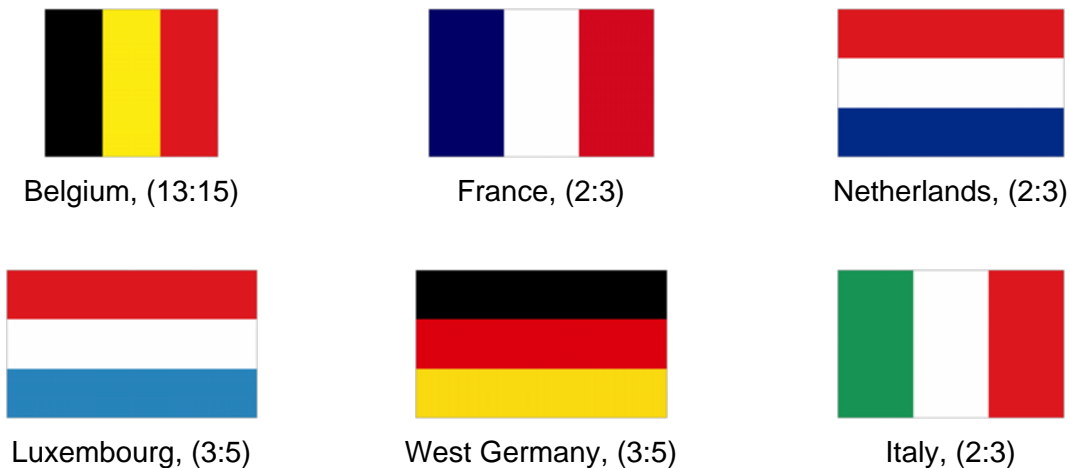


Figure 5. Flags of foundation states of EU

We can solve this problem in several levels of parameterization. Direct solution used by majority of competitors means one independent procedure for every flag. One competitive team used procedure with one parameter determining the state. After testing this parameter one sequence of commands were executed. They used only one procedure, but we can not talk about algorithm solving class of problems.

Decomposition of flags into two groups according to orientation of bands is the next level of problem analysis. We can create self procedure for every group. This solution was not used by any competitive team.

We can reach the highest level of parameterization by creating the only one procedure. Its parameters are orientation of bands (*orientation*), list of colours (*colours*) and length of flag sides (*width* and *height*). Only one competitive team scored this level.

```
to flag :orientation :colours :width :height
  right :orientation
  if :orientation=0 [right 90 forward :width left 90]
  let "height :height/3
  repeat 3 [setpencolour item repcount :colours
    polygon [forward :height
      left 90
      forward :width
      left 90
      forward :height]
    forward :height]
end
```

We can draw the flag of Belgium by `flag 90 ["black "yellow "red5] 182 210`, the flag of Netherlands by `flag 0 ["blue3 "white "red5] 210 140`.

## Conclusions

Based on two years experience with running PALMA junior competition we can say that we consider our effort as successful in some areas. The competitors were able to solve the competition problems and we believe that this competition helps them to increase their problem solving and programming skills and mathematical thinking. According to the teachers' opinions this competition helps them to (manage and) encourage learning and self-learning of gifted pupils in programming and also ordinary programming lessons because we have published study material, competition problems with assignments, solutions, commentaries and picture gallery on the competition website that is available for everyone.

We believe that our competition PALMA junior has its own place among other programming competitions in our region because of its approaches - covering aims in 4 dimensions/areas, on-line accessibility, openness to involving for everyone, publishing collection of all competition problems with results, commentaries.

In future we would like to continue in team style of organizing the competition and to create problems for competition covering mentioned aims and objectives using analytical approach. We intend to improve learning materials for the competition, evaluation tools for whole competition progress and also prepare on-line course for teachers focused on methodology of programming. In case of interest we are open to foreign competitors who can solve programming problems in English version of Imagine.

We would like to thanks our colleagues Gabriel Semanišin and Gabriela Andrejková for their support and advices and their help in the field of propagation of the competition and acquirement of prizes for finalists. We also thank to our students Alena Hajdová, Jana Brandoburová for their help in preparing assignment of problems and for checking competitors' solutions.

All results published in the article have been achieved in the frame of APVV project LPP-0131-06 "Increasing of knowledge potential".

## References

- Dagiene, V. (2005) *Competition in Information Technology: an Informal Learning*. In Proceedings of the Tenth European Logo Conference, Warsaw: Centre for Informatics and Technology in Education, 2005, pp. 228-234, ISBN 83-917700-8-7
- Dvorský, M. et al. (2007) *Palma – programming competition*. P. J. Šafárik University in Košice, Faculty of Science <<http://palma.strom.sk/>>
- Jochemczyk, W. and Katarzyna Oledzka, K. (2005) *Logo competition for primary school children*. In Proceedings of the Tenth European Logo Conference, Warsaw: Centre for Informatics and Technology in Education, 2005, pp. 383-385, ISBN 83-917700-8-7
- Knott, R. (2007) *Fibonacci Numbers and the Golden Section*. <<http://www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/>>
- Šlapkauskienė, V. (2005) *National Logo competition-Olympiad in Lithuania..* In Proceedings of the Tenth European Logo Conference, Warsaw: Centre for Informatics and Technology in Education, 2005, pp. 367-369, ISBN 83-917700-8-7
- Tomcsányiová, M. and Tomcsányi, P. (2005) *Logo programming competition in Slovakia*, In Proceedings of the Tenth European Logo Conference, Warsaw: Centre for Informatics and Technology in Education, 2005, pp. 377-382, ISBN 83-917700-8-7