Univerzita Pavla Jozefa Šafárika v Košiciach

Prírodovedecká fakulta

Tvorba úloh

pre programátorské súťaže

Ľubomír ŠNAJDER – Ján GUNIŠ

Názov: Tvorba úloh pre programátorské súťaže

Autori: RNDr. Ľubomír Šnajder, PhD., ÚINF PF UPJŠ v Košiciach

PaedDr. Ján Guniš, PhD., ÚINF PF UPJŠ v Košiciach

Vedecký redaktor: doc. RNDr. Stanislav Lukáč, PhD.

Recenzenti: doc. RNDr. Gabriela Lovászová, PhD.

RNDr. Róbert Hajduk, PhD.

Rozsah strán: 79

Rozsah AH: 4,4

Vydavateľ: Univerzita Pavla Jozefa Šafárika v Košiciach

Vydanie: prvé

Rok vydania: 2014

Dostupné od: 28. 04. 2014

Umiestnenie: http://www.upjs.sk/pracoviska/univerzitna-kniznica/e-publikacia/#pf

ISBN 978-80-8152-139-3

Táto monografia bola podporovaná Agentúrou na podporu výskumu a vývoja na základe Zmluvy č. APVV-0057-09 Rozvíjanie talentu prostredníctvom korešpondenčných seminárov a súťaží a Zmluvy č. APVV-0715-12 Výskum efektívnosti metód inovácie výučby matematiky, fyziky a informatiky.

AGENTÚRA
NA PODPORU
VÝSKUMU A VÝVOJA

Bibliografický odkaz:

ŠNAJDER, Ľubomír a Ján GUNIŠ. *Tvorba úloh pre programátorské súťaže* [online]. Košice: Prírodovedecká fakulta, UPJŠ v Košiciach, 2014, [cit. 2014-04-25]. ISBN 978-80-8152-139-3. Dostupné z: http://www.upjs.sk/pracoviska/univerzitna-kniznica/e-publikacia/#pf

ÚVOD 4
1 INFORMATICKÉ SÚŤAŽE NA SLOVENSKU 5
Olympiáda v informatike (OI) 6
ZENIT 7
Korešpondenčný seminár z programovania (KSP) 7
Imagine Logo Cup 8
ProFIIT – súťaž stredoškolákov v programovaní 8
Športové programovanie oči neodpudzujúcich grafických interaktívnych aplikácií (ŠPONGIA) 8
Programovanie, ALgoritmy, MAtematika (PALMA) 8
PALMA junior 9
IHRA 9
RoboCup junior 10
First Lego League (FLL) 10
Istrobot 11
Robotický Battle na Alejovej (RBA) 11
Súťaž Baltie 11
Internet Problem Solving Contest (IPSC) 12
Informatický Bobor (iBobor) 12
Virtuálna realita bez hraníc (VRBH) 12
Ďalšie súťaže 13
2 ÚLOHY V ŠKOLSKEJ INFORMATIKE 14
3 ANALÝZA VYBRANÝCH ÚLOH SÚŤAŽE PALMA JUNIOR 23
Atribúty súťažných úloh 23
Úloha o jesennom slniečku 26
Úloha o kreslení šachovnice 28
Úloha o tom, ako sa programuje vysávač 34
Úloha o súboji robotov 39
Úloha o šetrnom vykurovaní 46
Úloha o treskúcej zime a snehovej vločke 50
Úloha o prívesku z tvarovaného drôtika 55
Úloha o smere vetra 59
Úloha o trojskokanskej súťaži korytnačiek 64
Úloha o tom, ako korytnačky v bludisku blúdili 66
Úloha o tom, ako Korypolícia kontroluje dodržiavanie dopravných predpisov 72
ZÁVER 76
LITERATÚRA 77

ÚVOD

V monografii sú uvedené priebežné výsledky kvalitatívneho pedagogického výskumu autorov zameraného na tvorbu a evalváciu úloh z programovania, ktorý sa realizuje od roku 2005 na vlastnej programátorskej súťaži PALMA junior s podporou APVV projektov – APPV-0057-09 Rozvíjanie talentu prostredníctvom korešpondenčných seminárov a súťaží a APVV-0715-12 Výskum efektívnosti metód inovácie výučby matematiky, fyziky a informatiky.

Cieľom monografie je poskytnúť čitateľom prehľad informatických súťaží, problematiku tvorby úloh pre súťaže a tiež pre riadnu výučbu, na príklade vybraných úloh súťaže PALMA junior ukázať ako formulovať a preformulovávať zadania úloh, vytvárať prípravné (návodné) úlohy, analyzovať žiacke riešenia a zaznamenávať programátorské miskoncepcie žiakov.

Monografiu tvoria tri kapitoly: Informatické súťaže na Slovensku, Úlohy v školskej informatike, Vybrané úlohy súťaže PALMA junior. V prvej kapitole je uvedený prehľad informatických súťaži na Slovensku pre žiakov základných a stredných škôl – ich stručná charakteristika, organizátori, URL ich webovej stránky. V druhej kapitole je rozpracovaná problematika učebných úloh – ich komponentov, zamerania, typov formulácií, tvorby systémov úloh. Tretia kapitola obsahuje zadania a komentované autorské riešenia súťažných úloh reprezentujúcich zameranie súťaže na programovanie, algoritmy a matematiku. Pri každej úlohe je uvedená analýza žiackych riešení zameraná na spôsoby myslenia a typické miskoncepcie žiakov, súbory prípravných a rozširujúcich úloh.

Predložená monografia je určená didaktikom informatiky, autorom úloh programátorských súťaží a učiteľom informatiky. Radi privítame Vaše komentáre a názory k monografii na e-mailových adresách lubomir.snajder@upjs.sk, jan.gunis@upjs.sk.

Autori

1 INFORMATICKÉ SÚŤAŽE NA SLOVENSKU

Neodmysliteľnou súčasťou neformálneho (semi-formálneho) informatického vzdelávania sú informatické súťaže. Niektoré sú určené bežnej populácií žiakov, iné užšej skupine informaticky nadaných žiakov. Hlavným cieľom informatických súťaží včítane ich sprievodných podujatí (inštruktáži, sústredení, táborov) je umožniť žiakom prezentovať svoje odborné vedomosti, porovnať navzájom svoje vedomosti, zlepšiť informatické vedomosti a zručnosti, podporiť pozitívny vzťah k informatike, vytrvalosť, súťaživosť, atď.

Učiteľ informatiky môže využiť širokú paletu informatických súťaží na to, aby zapojil svojich žiakov do súťaží, ktoré im budú vyhovovať a budú prispievať k ich informatickému vzdelávaniu. Informatické súťaže sa líšia vo viacerých atribútoch:

- mierou zapojenia programátorských zručností – programátorské, neprogramátorské,
- mierou zapojenia sa účastníkov – medzinárodné, národné, lokálne,
- počtom súťažiacich (veľkosťou súťažného tímu) – jednotlivci, dvojice, n-tice (n>2),
- spôsobom zapojenia sa – prezenčné, dištančné (korešpondenčné, on-line),
- spôsobom vyhodnocovania riešení účastníkov (automatické strojové, ručné vrátane komentárov a pod.)
- sprievodnými podujatiami – inštruktáže, sústredenia, tábory,
- organizátorom – vysoká škola s informatickým zameraním, iná vzdelávacia inštitúcia.

Prehľad informatických súťaží:

Názov súťaže	Cieľová skupina	Úrovne	Programovací jazyk	Organizátor
OI	SŠ – jednotlivci	domáca, krajská, celoštátna, medzinárodná	Pascal, C, C++	MŠ SR, SISp, SKOI, JSMF
ZENIT	SŠ – jednotlivci	domáca, krajská, celoštátna	TurboPascal, FreePascal, Delphi	ŠIOV, COK ZENIT
KSP	SŠ – jednotlivci	celoštátna	Pascal, C++, Python	OZ Trojsten, FMFI UK
Imagine Logo Cup	ZŠ – jednotlivci	školská, celoštátna	Imagine Logo	FMFI UK
ProFIIT	SŠ – jednotlivci, dvojice	korešpondenčná + prezenčná celoštátna	Pascal, C, C++	FIIT STU
ŠPONGIA	SŠ – jednotlivci a	celoštátna	ľubovoľný jazyk	ŠpMNDaG

	tímy max 6-členné			Bratislava
PALMA	SŠ – dvojice, jednotlivci	domáca, online, celoštátna	Pascal, C, C++, Java	ÚINF PF UPJŠ
PALMA junior	SŠ – dvojice, jednotlivci	domáca, online, celoštátna	Imagine Logo	ÚINF PF UPJŠ
IHRA	ZŠ, SŠ, VŠ, iná – jednotlivci, tímy	celoštátna	ľubovoľný jazyk	ÚINF PF UPJŠ
RoboCup junior	ZŠ, SŠ (<19)– tímy	národná, medzinárodná	NXT-G, NXC	SSE, Robotika.SK
FLL	ZŠ, SŠ (10 – 16) – tímy	regionálna, kvalifikačná, stredo-EÚ, EÚ, celosvetová	NXT-G, ROBOLAB	NDS
Istrobot	bez vekového obmedzenia – tímy	celoštátna	ľubovoľný jazyk	ÚRaPI FEI STU, Robotika.SK
RBA	ZŠ, SŠ – tímy	celoštátna	ľubovoľný jazyk	G Alejová, Košice
Súťaž Baltie	ZŠ – jednotlivci	školská, okresná, krajská, národná, medzinárodná	SGP Baltík 3, SGP Baltie 4 C#	SGP Systems, OZ TIB
IPSC	bez vekového obmedzenia – tímy	celoštátna	–	FMFI UK
iBobor	ZŠ, SŠ – jednotlivci	celoštátna	–	FMFI UK
VRBH	ZŠ, SŠ – jednotlivci, tímy	celoštátna	–	FMFI UK

Postupne si predstavíme jednotlivé súťaže, najprv programátorské a potom neprogramátorské.

Olympiáda v informatike (OI)

URL = http://oi.sk/

OI je predmetová olympiáda pre stredoškolákov (jednotlivcov) v programovaní. Súťaž prebieha prezenčne v kategórii A v troch kolách – domáce, krajské, celoštátne, pričom najlepší riešitelia dostanú šancu reprezentovať Slovensko na medzinárodných programátorských súťažiach – Medzinárodnej olympiáde v informatike – International Olympiad in Informatics (IOI) http://www.ioinformatics.org/ a Stredoeurópskej olympiáde v informatike – Central European

Olympiad in Informatics (CEOI). V kategórii B sa súťaží v dvoch kolách – domácom a krajskom. Prvých 21 ročníkov OI prebehlo ako kategória P matematickej olympiády.

Súťaž sa organizuje od roku 1985. Vyhlasovateľmi OI sú Ministerstvo školstva, vedy, výskumu a športu Slovenskej republiky (MŠVVŠ SR) v garancii a spolupráci so Slovenskou informatickou spoločnosťou a Slovenskou komisiou Olympiády v informatike a v spolupráci s Jednotou slovenských matematikov a fyzikov. Sprievodnými podujatiami OI sú inštruktáže a sústredenia. Súťaž OI a jej organizačný poriadok sú registrované na MŠVVŠ SR pod číslom MŠSR-7683/2010-912.

Programovacie jazyky: Pascal, C, C++ (kategória A), ľubovoľný jazyk (kategória B).

ZENIT

URL = http://zenit.edu.sk/otpp.aspx

Súťaž v programovaní, elektronike a strojárstve stredoškolákov – jednotlivcov prebieha prezenčne v troch kolách – školské, krajské, celoštátne a v dvoch kategóriách – A (žiaci 3. a 4. ročníka SŠ), B (žiaci 1. a 2. ročníka SŠ). Hlavným cieľom súťaže je prispievať k vyhľadávaniu nadaných a talentovaných študentov, podporovať ich ďalší odborný rast, tvorivo rozvíjať kompetencie, viesť k samostatnej tvorivej činnosti a podporovať ich záujem o sebavzdelávanie.

Súťaž organizujú Štátny inštitút odborného vzdelávania Bratislava a Celoštátna odborná komisia ZENIT od roku 1984. Súťaž ZENIT programovaní a jej organizačný poriadok sú registrované na MŠVVŠ SR pod číslom 2010-11030/30509:8-913.

Programovacie jazyky: TurboPascal, FreePascal, Delphi.

Korešpondenčný seminár z programovania (KSP)

URL = http://www.ksp.sk/

KSP je súťaž stredoškolákov v riešení algoritmických úloh. Cieľom súťaže je zdokonaliť žiakov v programovaní a v algoritmickom myslení. Pozostáva z troch rôzne obťažných kategórií – Z (začínajúci), O (skúsenejší), T (špeciálna kategória pre náročných). V priebehu školského roka sa organizuje zimná a letná časť súťaže, každá z nich má dve kolá. Účastník v každom kole rieši v svojej kategórii 5 súťažných úloh. Kódy programov sa testujú automaticky, popisy algoritmov riešení a ich zložitosti sa opravujú ručne.

Organizátormi KSP sú OZ Trojsten a FMFI UK v Bratislave od roku 1983. Sprievodnými podujatiami súťaže sú sústredenia pre najúspešnejších riešiteľov.

Programovacie jazyky: Pascal, C++, Python.

Imagine Logo Cup

URL = http://edi.fmph.uniba.sk/~tomcsanyiova/ImagineLogoCup/

Imagine Logo Cup je súťaž v programovaní pre jednotlivcov – žiakov 6. – 9. ročníka ZŠ a prímy až kvinty osemročných gymnázií. V školskom kole v priebehu 90 minút každý žiak rieši tri zadania. Víťazi školského kola postupujú na celoslovenské kolo.

Organizátorom súťaže je Katedra základov a vyučovania informatiky FMFI UK v Bratislave s podporou Slovenskej informatickej spoločnosti od roku 1997.

Programovacie jazyky: Imagine Logo.

ProFIIT – súťaž stredoškolákov v programovaní

URL = http://www2.fiit.stuba.sk/ProFIIT/

ProFIIT je celoslovenská programátorská súťaž pre jednotlivcov a dvojice stredoškolákov. Súťažiaci v rámci korešpondenčnej časti súťaže zasielajú svoje riešenia cez internet. Najlepší súťažiaci postupujú do finálovej prezenčnej časti súťaže. Víťazi a úspešní riešitelia vo finále ProFIIT získajú body do hodnotenia prijímacieho konania na FIIT STU v Bratislave pre nastávajúci akademický rok.

Súťaž organizuje FIIT STU v Bratislave od roku 2004.

Programovacie jazyky: Pascal, C, C++.

Športové programovanie oči neodpudzujúcich grafických interaktívnych aplikácií (ŠPONGIA)

URL = http://www.smnd.sk/main/spongia/

ŠPONGIA je celoslovenská súťaž v programovaní počítačových hier pre žiakov stredných škôl. Súťažia 1-6 členné tímy, ktoré majú organizátorom v priebehu 17 dní od vyhlásenia poslať hru na zadanú tému (spustiteľný a zdrojový kód, manuál pre používateľa, základné informácie o hre).

Súťaž organizuje Škola pre mimoriadne nadané deti a gymnázium, Teplická 7, v Bratislave od roku 2006.

Programovacie jazyky: ľubovoľné.

Programovanie, ALgoritmy, MAtematika (PALMA)

URL = http://palma.strom.sk/

Programátorská súťaž PALMA je určená pre žiakov stredných škôl z celého Slovenska – dvojice alebo jednotlivcov. V každom školskom roku sa konajú dve domáce kola, v ktorých majú

súťažiaci v priebehu troch hodín vyriešiť 4 – 5 úloh. Najúspešnejší riešitelia z domácich kôl postúpia do prezenčného finálového kola, ktoré prebieha 2 súťažné dni. V tejto súťaži sa odovzdávajú len hotové programy, ktoré sú vyhodnocované automaticky, t. j. sú spustené s viacerými súbormi vstupných dát, ktoré súťažiaci nepoznajú.

Súťaž organizuje Ústav informatiky Prírodovedeckej fakulty UPJŠ v Košiciach.

Programovacie jazyky: Pascal, C, C++, Java.

PALMA junior

URL = http://di.ics.upjs.sk/palmaj/

PALMA junior je súťaž v programovaní pomocou programovacieho prostredia Imagine pre žiakov druhého stupňa základných škôl, žiakov prímy až sexty osemročných gymnázií a žiakov 1. a 2. ročníka stredných škôl. Hlavným cieľom súťaže je povzbudiť žiakov, aby riešili zaujímavé algoritmické problémy, zlepšili si svoje programátorské zručnosti a matematické myslenie. Do súťaže sa môžu prihlásiť dvojčlenné tímy žiakov alebo jednotlivci. Súťažiaci sú rozdelení do dvoch vekových kategórií – PROFÍK (žiaci 5. – 7. ročníka ZŠ) a EXPERT (žiaci 8. – 9. ročníka ZŠ a 1. – 2. ročníka SŠ). V každom školskom roku sa organizujú tri on-line školské kolá a jedno finálové kolo, do ktorého postupujú najúspešnejší riešitelia školských kôl. V školských kolách každý tím v priebehu 3 hodín rieši 4 úlohy, ktorých riešenia sa odosielajú na server organizátorov súťaže.

Súťaž organizuje Ústav informatiky Prírodovedeckej fakulty UPJŠ v Košiciach od roku 2005. Od školského roku 2011/2012 sú súťaž PALMA junior a jej organizačný poriadok registrované na MŠVVŠ SR pod číslom 2011-63/7940:31-922.

Programovacie jazyky: Imagine Logo.

IHRA

URL = http://web.ics.upjs.sk/ihra/

IHRA je súťaž jednotlivcov a tímov v programovaní počítačových hier pre žiakov základných a stredných škôl. vysokoškolských študentov a iných počítačových nadšencov. Cieľom súťaže je priviesť žiakov k tvorbe vlastných počítačových hier a rozvoju ich programátorských schopností, tvorivosti a estetického cítenia. V priebehu 3 mesiacov (január – marec) majú súťažiace tímy zaslať organizátorovi súťaže ľubovoľnú hru (spustiteľný kód a popis hry) so zakomponovaným logom UPJŠ a prezentovať ju na pôde organizátora pred hodnotiacou komisiou a ďalšími súťažiacimi.

Súťaž organizuje Ústav informatiky Prírodovedeckej fakulty UPJŠ v Košiciach od roku 2012.

Programovacie jazyky: ľubovoľné.

RoboCup junior

URL = http://robotika.sk/rcj/

RoboCup junior je súťaž v stavbe a programovaní robotov pre základné a stredné školy (veková hranica 19 rokov). Súťažiace tímy sa môžu zapojiť do niektorých zo štyroch kategórií (Konštrukcia, Robotický futbal, Záchranár, Tanec). Cieľom súťaže je umožniť súťažiacim zrealizovať svoje nápady pri konštrukcii a programovaní robotov, podporiť ich kreativitu a prácu v tíme, prípadne sa zoznámiť s ďalším programovacím jazykom pre robotické modely.

Súťaž na Slovensku organizuje Slovenská Spoločnosť Elektronikov v spolupráci so združením Robotika.SK od roku 2003. Najúspešnejší súťažiaci národného kola súťaže v troch kategóriách Robotický futbal, Záchranár, Tanec postúpia na medzinárodné kolo súťaže RoboCup junior (http://rcj.robocup.org/).

Programovacie jazyky: NXT-G, NXC.

First Lego League (FLL)

URL = http://www.fll.sk/

FLL je robotická súťaž pre tímy žiakov základných škôl vo veku 10 až 16 rokov. Cieľom súťaže je preveriť logické myslenie pri riešení problémov, schopnosti v programovaní robotov v ikonografických jazykoch, upozorniť na možné výskumné úlohy v okolí súťažiacich, rozvíjať tímovú spoluprácu a kreativitu súťažiacich. Súťažné tímy postavia roboty výlučne zo stavebníc LEGO použitím základných senzorov a naprogramované štandardným softvérom. Jednotlivé súťažné tímy sa zúčastnia súťažného turnaja, na ktorom ich doma naprogramované roboty plnia na ihrisku FLL rôzne úlohy, za ktoré získavajú body. Navyše tímy riešia aj výskumnú úlohu vo svojom okolí, ktorú prezentujú na turnaji.

Súťaž prebieha najprv v regionálnych turnajoch v piatich slovenských mestách, úspešné tímy z týchto turnajov postúpia do kvalifikačných semifinálových turnajov. Ich víťazi sa dostanú na stredoeurópske finále (FLL Finale Central Europe), ktoré organizuje od roku 2001 nadácia HANDS on TECHNOLOGY (http://www.hands-on-technology.de/en/firstlegoleague/). Napokon ich víťazi postúpia na FLL Open European Championship a World Festival.

Súťaž FLL sa koná od roku 2007. Organizáciu súťaže prevzala roku 2010 Nadácia pre deti Slovenska (http://www.nds.sk/), ktorá tiež organizuje bezplatný tréning trénerov. Od školského roku 2011/2012 sú súťaž FLL a jej organizačný poriadok registrované na MŠVVŠ SR pod číslom 2011-63/41451:52-922.

Programovacie jazyky: NXT-G, ROBOLAB.

Istrobot

URL = http://www.robotika.sk/contest/

Istrobot je súťaž tímov alebo jednotlivcov bez vekového obmedzenia v programovaní robotických modelov. Súťaží sa v štyroch kategóriách – Stopár, Myš v bludisku, V sklade kečupu, Voľná jazda. Pred súťažou je možné si otestovať model počas dvoch testovacích dní na ihriskách organizátora. Každý súťažiaci tím pred súťažou poskytne dokumentáciu popisujúcu elektroniku, konštrukciu a riadiaci algoritmus svojho robotického modelu.

Súťaž organizuje Ústav riadenia a priemyselnej informatiky FEI STU v spolupráci so združením Robotika.SK od roku 2000. Sprievodným podujatím súťaže je robotický workshop pre účastníkov z krajín Višegrádskej štvorky.

Programovacie jazyky: ľubovoľné, závislé od použitého materiálu na konštrukciu robota.

Robotický Battle na Alejovej (RBA)

URL = http://www.galeje.site90.net/index.php?clanok=rba/a-o_sutazi

RBA je robotická súťaž pre žiakov základných a stredných škôl. Súťaž prebieha v troch kategóriách: Vlastný model – Model z praxe, RobotSolve, Racing. Cieľom súťaže je predstavenie robotiky a informatiky na školách a jej zviditeľnenie.

Súťaž organizuje Gymnázium, Alejová 1 v Košiciach.

Programovacie jazyky: ľubovoľné.

Súťaž Baltie

URL = http://baltie.net/(S(2grg0lsnyadwyymh5iu1fegi))/

Baltie je súťaž v programovaní pre žiakov základných škôl. Súťaží sa v troch kategóriách A (0. – 3. ročník), B (4. – 6. ročník), C (7. – 9. ročník). Pred rokom 2013 bola Baltie súťažou tímov, odteraz je súťažou jednotlivcov. Súťažiaci riešia úlohy školského kola a najúspešnejší sa môžu dostať postupne do okresného, krajského, národného až medzinárodného kola súťaže Baltie. V každom kole súťažiaci riešia 5 úloh, ktorých zdrojové kódy riešení posielajú na server súťaže na automatické vyhodnotenie.

Súťaž organizuje SGP Systems (http://www.sgpsys.com/) s partnermi, na Slovensku je to Občianske združenie Tvorivá informatika s Baltíkom (http://www.tib.sk/).

Programovacie jazyky: SGP Baltík 3 a SGP Baltie 4 C#.

Internet Problem Solving Contest (IPSC)

URL = http://ipsc.ksp.sk/

IPSC je on-line súťaž v riešení problémov pre maximálne trojčlenné tímy súťažiacich. Súťažiacim sa predloží 10 až 20 problémov (zadania a vstupné súbory), na riešenie ktorých majú 5 hodín. Súťažiaci k jednotlivým problémom zasielajú výstupné súbory, nie programy, ktoré riešia zadané problémy. Je dovolené použiť ľubovoľne verejné dostupné informačné zdroje a nástroje (napr. Google, Wikipedia, MathWorld, WolframAlpha). Odporúča sa deň pred súťažou zúčastniť sa tréningového on-line spojenia, aby sa súťažiaci oboznámili s prostredím súťaže. Súťaž prebieha v anglickom jazyku.

Súťaž organizujú informatické katedry FMFI UK v Bratislave od roku 1999.

Informatický Bobor (iBobor)

URL = http://ibobor.sk/

IBobor je veľmi populárna on-line informatická súťaž pre jednotlivcov – žiakov od 3. ročníka základných škôl až po maturitu. Súťaží sa v piatich vekových kategóriách (Bobríci, Benjamíni, Kadeti, Juniori, Seniori). Každý zaregistrovaný súťažiaci má na vyriešenie 15 úloh čas 40 minút, v najmladšej kategórii Bobríci je to 12 úloh a čas 30 minút. Zadávané úlohy majú rôznu podobu – výber odpovede, interaktívne úlohy (presúvanie, doplnenie, usporadúvanie objektov). Úlohy svojím zameraním patria do niektorej z oblasti: Fakty, Algoritmizácia a programovanie, Porozumenie informáciám, Logika, Praktické a technické otázky, IKT v každodennom živote, Matematické základy informatiky.

Pre učiteľov je k dispozícii Bobrovo učiteľské prostredie, v ktorom majú k dispozícii všetky úlohy iBobra, z ktorých môžu vytvárať vlastné testy pre svoju výučbu informatiky (http://www.st.fmph.uniba.sk/~hrusecky1/ibobor_testy/ucitel/indexUcitel.php).

Súťaž organizuje Katedra základov a vyučovania informatiky FMFI UK v Bratislave od roku 2007.

Virtuálna realita bez hraníc (VRBH)

URL = http://www.sccg.sk/vrbh/

VRBH je súťaž jednotlivcov a tímov od 10 rokov po maturitu v tvorbe prózy alebo poézie zameranej tematicky na matematiku, informatiku, fyziku a iné predmety. Súťažiaci pod vedením tútorov preukážu, že vedia tvorivo a kriticky spracovať niektorú s predložených tém. Súťažné príspevky sa prezentujú a vyhodnotia na minikonferencii, súčasťou ktorej sú prednášky a workshopy

učiteľov a doktorandov FMFI UK v Bratislave zamerané na populárno-vedecké témy z matematiky, fyziky, informatiky.

Súťaž organizuje FMFI UK v Bratislave od roku 2010.

Ďalšie súťaže

V období rokov 2006 až 2008 organizovala FMFI UK v Bratislave súťaž **HALUZ** (http://haluz.org/) pre nanajvýš 5-členné tímov v riešení šifier a logických úloh. Súťaž prebiehala vo viacerých (napr. 6) kolách. Do celkového poradia sa každému tímu zarátali tri jeho najlepšie výsledky.

Medzi špičkové svetové programátorské súťaže patria **TopCoder** (http://www.topcoder.com/) a **Google Code Jam** (https://code.google.com/codejam/), do ktorých sa zapájajú skôr vysokoškoláci.

Prehľad informatických súťaží uvádza tiež Julka Šišková na webovej stránke: http://people.ksp.sk/~julka/sutaze/.

Námety prieskumných otázok pre učiteľov:

- Zistite v triede, do ktorých informatických súťaží sa zapájajú vaši žiaci. Zapájajú sa títo žiaci aj do iných súťaží, napr. matematických?
- Zistite, z akých dôvodov sa zapájajú vaši žiaci do informatických a iných súťaží.
- Poznajú vaši žiaci informatické súťaže, ktoré sa organizujú na Slovensku? Ak nie, poskytnite im stručný prehľad informatických súťaží a pomôžte sa im zapojiť do vybraných súťaží.
- Zapájali ste sa počas svojho štúdia do riešenia informatických alebo iných predmetových súťaží? Aký máte vzťah ku informatickým súťažiam – máte chuť a viete si nájsť čas na koordinovanie a odbornú pomoc pre vašich žiakov zapojených do informatických súťaží?
- Chápete zapojenie svojich žiakov do súťaží ako ďalšiu možnosť pre nich sa rozvíjať v informatike aj mimo vašej výučby informatiky za pomoci iných ľudí, ktorí organizujú informatické súťaže?
- Aký záujem o účasť žiakov na informatických súťažiach a sprievodných podujatiach majú rodičia, vedenie školy? Ako k tomu prispievajú?

2 ÚLOHY V ŠKOLSKEJ INFORMATIKE

Základným stavebným kameňom informatických súťaží sú **úlohy**, prostredníctvom ktorých sa napĺňajú ciele týchto súťaží. Učebné úlohy sú prostriedkom aktivizácie žiakov, spätnej väzby a riadenia výučby. V rámci svojich plánovacích a didaktických kompetencií by mal učiteľ informatiky byť schopný zostavovať úlohy (resp. systémy úloh) pokrývajúce vybrané špecifické ciele, správne formulovať zadania úloh, vytvárať k daným úlohám analogické/alternatívne, pomocné, prípadne rozširujúce úlohy.

Podľa (ŠVEDA, 1992) pod učebnou úlohou rozumieme každú **požiadavku** na **merateľnú činnosť žiaka** zameranú **na dosiahnutie cieľov učenia sa**. Podľa D. Holoušovej (KALHOUS – OBST, 2002) definujeme učebné úlohy ako širokú škálu všetkých učebných zadaní, a to od najjednoduchších úloh vyžadujúcich pamäťovú reprodukciu poznatkov, až po zložité úlohy vyžadujúce tvorivé myslenie.

Učebná úloha môže mať rôzne podoby, napr. cvičenie, problém, projekt, otázka, diskusia. (GUNIŠ – ŠNAJDER, 2009).

Cvičenie považujeme za úlohu vyžadujúcu rutinné riešenie. Z pohľadu žiaka mu dáva možnosť precvičiť si a upevniť prebrané učivo – napr. úloha na vykreslenie zadaného počtu schodov pomocou korytnačej grafiky.

Problémom je úloha, ktorá nemá prvoplánové riešenie a vyžaduje si využitie vyšších kognitívnych funkcií a rôznych stratégií riešenia – napr. úloha na určenie minimálneho počtu bankoviek a mincí pre zadanú sumu a zadané druhy bankoviek a mincí (mincovka).

Za **projekt** považujeme úlohu vyžadujúcu samostatné riešenie žiaka, ktorá má praktické použitie – napr. naprogramovanie hry, v ktorej počítač háda nami myslené prirodzené číslo zo zadaného intervalu.

Úlohou môže byť aj **otázka** zisťujúca žiakove vedomosti – napr. *Koľko otázok typu áno/nie potrebuješ položiť na zistenie myslenej karty z balíka 32 kusov nemeckých kariet?*

Úloha môže mať podobu **diskusie**, v ktorej žiaci preukážu jednak hĺbku pochopenia problému, jednak svoje schopnosti argumentovať, polemizovať, klásť otázky, atď. – napr. diskusia žiakov k riešiteľnosti problému vykreslenia n-cípej hviezdy.

V učebniciach a zbierkach úloh sa môžeme stretnúť aj s pojmom **príklad** (niekedy nesprávne označovaný ako riešený príklad), ktorý pozostáva zo zadania a riešenia úlohy („dva v jednom“). Jeho účelom je ukázať a vysvetliť žiakovi (čitateľovi) spôsob riešenia daného problému pomocou nových nástrojov (príkazov programovacieho jazyka), či postupov (algoritmov, stratégií riešenia problémov) –

napr. vyriešená úloha na prevod čísla z desiatkovej do dvojkovej sústavy. Príklad samotný neumožňuje zistiť mieru ako žiak porozumel riešeniu úlohy, preto odporúčame k príkladom doplniť kontrolné otázky.

V súťaži majú rozhodujúce postavenie problémy. Podľa (KOPKA, 2006) problém pozostáva z troch zložiek:

1. Východisková situácia, v ktorej popisujeme súvislosti a poskytujeme informácie alebo dáta.
2. Cesta od východiskovej situácie k cieľu, ktorá pre riešiteľa môže ale aj nemusí byť zrejmá alebo dosiahnuteľná.
3. Cieľ (koncová situácia), ktorý chce riešiteľ dosiahnuť.

Podľa úrovne presnosti popisu jednotlivých zložiek (KOPKA, 2006) rozlišuje nasledovné typy problémov:

- **Rutinný problém (cvičenie)** – východisková situácia je presne popísaná, cieľ je presne zadaný a cesta k nemu je známa,
- **Skutočný problém** – východisková situácia je presne popísaná, cieľ je presne zadaný, ale cesta k nemu nie je známa,
- **Skúmanie** – východisková situácia je presne popísaná, cieľ nie je presne zadaný alebo nie je zadaný vôbec a cesta k cieľu teda tiež nemôže byť známa.

Hranica medzi rutinným a skutočným problémom je závislá od riešiteľa problému. To čo predstavuje pre menej skúseného žiaka skutočný problém, môže byť pre skúsenejšieho žiaka rutinný problém. V súťaži PALMA junior sú úlohy formulované ako skutočné problémy.

Úlohy môžeme klasifikovať aj z pohľadu náročnosti ich riešenia. (LUKÁČ, 1999) rozlišuje tri typy úloh:

- **Elementárne** – riešiteľné využitím definícií príkazov a dátových typov. Úlohy tohto typu odrážajú úroveň osvojenia základných poznatkov.
- **Algoritmické** – zamerané na pochopenie súvislostí medzi základnými poznatkami. Vyžadujú presnú formuláciu a zápis známych alebo vysvetlených postupov na vyriešenie úlohy v algoritmickom jazyku. Vyznačujú sa vysokou určenosťou jednotlivých krokov.
- **Heuristické** – predstavujú aplikáciu nadobudnutých poznatkov na nové problémy a nové oblasti ľudskej činnosti. Sú to úlohy tvorivého charakteru, ktoré determinujú s menšou určitosťou vykonávanie jednotlivých krokov. Riešenie týchto úloh spočíva v ich rozložení na čiastkové úlohy a úplné vyriešenie úlohy nemusia dosiahnuť všetci žiaci.

V súťaži PALMA junior sú úlohy formulované ako heuristické.

Námety úloh a prieskumných otázok pre učiteľov a autorov úloh:

- Rozlišujete vo svojej učiteľskej praxi pojmy príklad (ako vzor, ilustráciu, ukážku) a úlohu (ako výzvu k riešeniu)?
- Aký zmysel má rozlišovať rôzne podoby úloh? Aké podoby úloh rozlišujete a používate v svojej školskej praxi?
- Uveďte príklad otázky, ktorá nie je úlohou (nesúvisí s cieľom vyučovania, nie je merateľná)?
- Pre každú z uvedených podôb úloh uveďte ďalšie konkrétne námety zadaní úloh.

Podľa (ŠVEC – FILOVÁ – ŠIMONÍK, 2004) má každá úloha tri kľúčové parametre:

- **Stimulačný** (motivačný) parameter – slúži na navodenie záujmu o poznávanie. Často v zadaniach úloh ide o vyrozprávanie reálneho, či fantazijného príbehu, či popisu zaujímavej situácie. Stimulačnú časť zadania úlohy vieme naformulovať tak, aby úloha rozvíjala aj afektívnu zložku žiaka (napr. postoje k rodine, škole, spoločnosti), medzipredmetové vzťahy (napr. s fyzikou, biológiou, občianskou náukou) a i.
- **Operačný** parameter – slúži na navodenie a rozvíjanie učebných operácií, odpovedajúcich vzdelávacím cieľom. Žiak by sa zo zadania úlohy mal dozvedieť čo má spraviť, čo sa od neho očakáva. V operačnej časti zadania by mali byť v druhej osobe uvedené pokyny pre žiaka. Extrémnym prípadom zadania bez motivačného parametra sú zadania, napr. „Mincovka“, „Kvadratická rovnica“, „Hádaj číslo“, ktoré sa dajú považovať za stručný názov úlohy, nie za plnohodnotné zrozumiteľné zadanie.
- **Regulačný** parameter – regulačnú silu úlohy ovplyvňuje jej určenosť, t. j. presnosť určenia jej zložiek, heuristickosť úlohy. Pomocou formulácie úlohy vieme ovplyvniť, či sa na jej riešenie budú od žiaka vyžadovať len nižšie, či aj vyššie kognitívne funkcie. Pomocou regulačného parametra nastavujeme náročnosť úlohy. Napr. v úlohe: *Vydláždi tehličkami celú miestnosť v ktorej stojí robot Karel* zvážime, ktoré z nasledovných podmienok uvedieme do zadania úlohy – počiatočnú (prípadne aj konečnú) polohu robota v miestnosti, rozmery miestnosti, mieru naplnenosti miestnosti tehličkami.

Námety úloh a prieskumných otázok pre učiteľov a autorov úloh:

- Nájdite v učebniciach programovania a na webových stránkach programátorských súťaží úlohy, ktoré majú rôznu úroveň stimulačného parametra (žiadnu, umelú a menej prirodzenú, zaujímavú a autentickú).

- Používate vo výučbe úlohy bez stimulačného parametra („nahé úlohy“)? Ak áno, v akých situáciách?
- Obľubujú vaši žiaci úlohy s rozvinutým stimulačným parametrom (podobne ako pri výučbe matematiky slovné úlohy)? Ako často zaraďujete do výučby takého úlohy?
- Ako rozsiahle stimulačné texty v úlohách sú pre vašich žiakov prijateľné?
- Do akej miery sa vo svojej výučbe zaoberáte rozborom zadaní úloh? Ukazujete žiakom ako vy rozmýšľate pri pochopení zadania úloh, s čím môžu byť problémy, ako ich prekonávať?
- Majme zadanie úlohy: *Vypočítanie NSD dvoch čísel*. Aké chyby ste zaregistrovali v tomto zadaní? Preformulujte toto zadanie tak, aby neobsahovalo zistené chyby.
- Naformulujte zadanie úlohy s názvom „Hádaj číslo“, aby bolo zrozumiteľné a obsahovalo zreteľné pokyny pre žiaka.
- Naformulujte dve zadania úlohy na vykreslenie n-cípej hviezdy, prvé čo najjednoduchšie (vyžadujúce nižšie kognitívne funkcie) a druhé čo najnáročnejšie (vyžadujúce vyššie kognitívne funkcie).

Viacerí autori vo svojich publikáciách odporúčajú sa zamerať nielen na samotné úlohy, ale na vytváranie premyslených **súborov (systémov) učebných úloh** (ŠVEDA, 1992) (KALHOUS – OBST, 2002).

Pri zostavovaní systémov úloh pre riadnu výučbu alebo pre programátorskú súťaž odporúčame zobrať do úvahy viaceré aspekty:

- Pokrývať **kognitívne, afektívne** a prípadne aj **psychomotorické ciele** výučby vybranej témy. Pre zadanú tému treba mať zmapované kognitívne ciele, ktoré by mali byť pokryté systémom úloh (pomôckou môže byť dvojrozmerná tabuľka s cieľmi a úlohami). Afektívne ciele sa dajú pokryť úpravou stimulačnej časti zadania úlohy. Psychomotorické ciele sú v školskej informatike sekundárne, dajú sa rozvíjať hlavne pri problematike spracovania multimédií, kde sa vyžaduje jemná motorika a koordinácia pohybov.
- Pri formulácii úloh orientovať sa tiež na rozvíjanie **vyšších kognitívnych funkcií** žiakov. Dobrou pomôckou sú nasledovné typy formulácií úloh zohľadňujúce pokrytie kognitívnych cieľov podľa Revidovanej Bloomovej taxonómie (RBT):
    - „Vytvorte/naprogramujte riešenie problému ...“, „Vytvorte program, ktorý ...“ – aplikovať, tvoriť (3. a 6. úroveň RBT).
    - „Modifikujte program, aby riešil ..., t. j. doplňte na vyznačené prázdne miesta programový kód, upravte/zmeňte/nahraďte časť programového kódu ...“ – aplikovať, analyzovať (3. a 4. úroveň RBT).

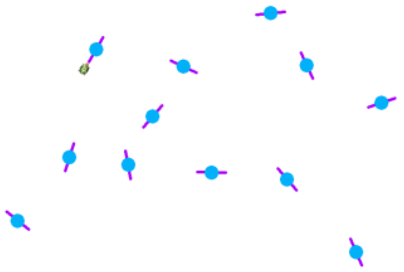
- „Analyzujte riešenie problému“, „Vysvetlite čo robí uvedený program“, „Nájdite (a opravte) chybu v programe“ – analyzovať (4. úroveň RBT).
- „Porovnajte a vyhodnoťte (podľa zadaného kritéria) uvedené riešenia“ – hodnotiť (5. úroveň RBT).

Námety na ďalšie formulácie zadaní úloh zohľadňujúce Fullerovu adaptáciu Bloomovej taxonómie vzdelávacích cieľov nájdeme v publikácii (MIKOLAJOVÁ, 2012) – „Ako na to?“, „Skúšaj sám – ...“, „Uprav program“, „Čo bude vykonávať program?“, „Dokonči projekt“, „Vylepši projekt“, atď.

- Zaradiť úlohy s **gradovanou náročnosťou** zohľadňujúc štádiá poznávacieho procesu (s použitím jediného, alebo viacerých rôznych kontextov). Ak použijeme metaforu „schodiska poznania“, každá vyriešená úloha systému úloh nás posunie o schodík vyššie na „schodisku poznania“. Príliš vysoké schody (t. j. príliš náročné úlohy) žiaka znechutia a demotivujú riešiť ďalšie úlohy. Príliš veľa schodov môže žiaka unaviť. Veľmi plytké schody môžu znechutiť hlavne šikovných žiakov, pre ktorých sú náročnejšie úlohy výzvou a aj nevyhnutnosťou, aby sa naučili riešiť problémy.
- Zohľadniť etapy vyučovacieho procesu a do systému úloh zaraďovať úlohy so zodpovedajúcimi **didaktickými funkciami**:
  - **motivačná** (na upútanie študentov a zvýšenie záujmu o danú problematiku – ukážka (takmer) hotovej počítačovej hry, demonštračné video navodzujúce potrebu naprogramovať demonštrovaný problém, uvedenie prípadovej štúdie),
  - **expozičná** (na prvotné osvojenie poznatkov – rutinné jednoduché úlohy, napr. na uvedenie konceptu cyklu; prípravné – pomocné úlohy predchádzajúce novému konceptu, napr. jednoduchá úloha na nekonečnú rekurziu; propedeutické – príprava konceptu v zjednodušenej podobe, napr. výpočet konkrétnej časovej náročnosti programu – počet použitých príkazov),
  - **fixačná** (upevňovacie – rutinné úlohy na precvičenie a prehĺbenie konceptu, napr. úlohy na cykly a vnorené cykly; proti stereotypu – pre odstránenie známej miskoncepcie, napr. pri preberaní cyklov typu kým/while zaradiť úlohu na podmienený príkaz ak/if; systemizačné – na zaradenie nového konceptu do štruktúry vedomosti žiaka, napr. úloha na cykly obsahujúce podmienené príkazy, ktoré sa preberali pred cyklami),
  - **diagnostická** (na preverenie kvality získaných vedomostí žiakov – pri bežnej výučbe ide skôr o zmeranie výkonu žiaka vzhľadom k zadanému kritériu – overujúce testovanie/test absolútneho výkonu; pri súťažiach ide hlavne o porovnanie jeho výkonu s inými žiakmi – rozlišujúce testovanie/test relatívneho výkonu; pri súťažných

zadaniach a projektoch nemôžeme byť silno upätí na autorské riešenie a hodnotiť výkon žiaka len vzhľadom k autorskému riešeniu, ale musíme brať do ohľadu aj iný prístup k riešeniu, napr. pri výpočte súčtu prvých n prirodzených čísel očakávame použitie cyklu, ale matematicky zdatný žiak namiesto cyklu použije vzorec $\frac{n(n+1)}{2}$),

- **aplikačná** (riešenie problémov zo života pomocou získaných vedomostí – programovanie modelov z reálneho života – skener, radar, tachometer, záznamník, šifrátor, pantograf, kaleidoskop, cyklopočítač ...).



Pri tvorbe a výbere úloh pre riadnu výučbu, resp. pre programátorskú súťaž musíme zohľadniť aké kognitívne ciele žiaka má pokrývať vybraný tematický celok, resp. súťaž. Ukážky pokrytia kognitívnych cieľov úlohami pre vybraný celok (dátový typ pole) sú uvedené v publikácii (GUNIŠ – ŠNAJDER, 2009) a pre súťaž PALMA junior v publikácii (GUNIŠ et al., 2007).

Ukážkou zaradenia stimulačného, operačného aj regulačného parametra úlohy, pokrytia kognitívnych a tiež afektívnych cieľov, vzťahov s inými predmetmi, rozvoja kritického a divergentného myslenia je úloha o cukríkoch, ktorá je modifikáciou úlohy zo súťaže PALMA junior (GUNIŠ – ŠNAJDER, 2013).

Mamka kúpila svojim šikovným detičkám cukríky. Keď prídu zo školy, chce ich prekvapiť a pekne im usporiadať cukríky na stolík. Nie tak neporiadne, ako je to na obrázku.

Popíš postup, pomocou ktorého môžeme vykresliť cukríky v ľubovoľnom, ale pravidelnom usporiadaní.

Nezabudni na to, že mamky môžu mať rôzne počty detí. Čo si myslíš, dá sa podľa uvedeného obrázka s istotou povedať, koľko mala mamka detí?

Námety úloh a prieskumných otázok pre učiteľov a didaktikov informatiky:

- Analyzujte zadanie uvedenej úlohy a vymenujte afektívne ciele, ktoré pokrýva táto úloha.
- Napadli vás aj iné ciele ako: estetické vykreslenie obrázku, zmysel pre poriadok, pozitívny vzťah matky a detí, zmysel pre spravodlivosť pri delení cukríkov?
- Vyriešte túto úlohu aspoň troma rôznymi spôsobmi. Podarilo sa vám to? Ktoré z riešení sa vám najviac páči? Aký druh myslenia rozvíjame u žiakov, keď im zadávame úlohy, ktoré majú viacero riešení?
- Aké vedomosti z matematiky je potrebné využiť pri riešení tejto úlohy?

- Uveďte v čom sa odlišujú žiaci, ktorí riešili túto úlohu nasledovne:
    - jeden uviedol v svojom riešení počet detí 2, 3, 4, 6,
    - druhý uviedol v svojom riešení počet detí 1, 2, 3, 4, 6, 12 (všetky delitele čísla 12),
    - tretí napísal, že deti mohlo byť hocikoľko, keď mamka nedala každému dieťaťu rovnaký počet cukríkov,
    - štvrtý napísal, že sa to nedá určiť, lebo aj keď vidíme na obrázku 12 rôznych cukríkov, v skutočnosti sa môžu cukríky prekrývať a môže byť ľubovoľný počet väčší alebo rovný 12.
- Vedeli by ste túto úlohu vyriešiť aj bez použitia programovacieho jazyka? Ktoré (softvérové) nástroje by ste použili? Využíva žiak algoritmické myslenie aj pri použití týchto nástrojov?

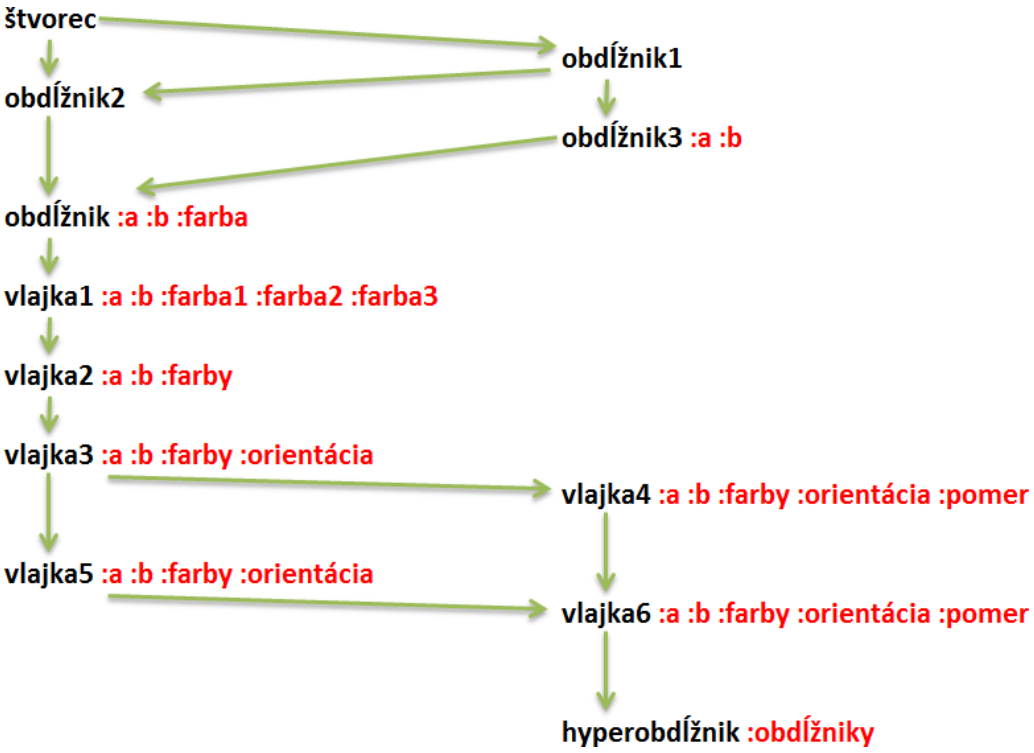
Ak je žiakovi predložená náročnejšia úloha je veľmi dôležité, aby poznal repertoár **stratégií riešení problémov** (PROJECT MATH, 2010) a vedel vybrať a použiť vhodné stratégie riešenia problému (napr. rozloženie problému do podproblémov, prevedenie problému na známy problém, nájdenie vzoru, riešenie problému od konca, použitie tabuľky, nakreslenie obrázka, odhad – overenie – oprava riešenia, atď.) Užitočnými stratégiami podľa (KOPKA, 2006) sú konkretizácia a zovšeobecnenie. Konkretizácia (napr. pri tvorbe prípravných úloh) spočíva vo vyriešení jedného alebo viacerých konkrétnych prípadov. Zmyslom konkretizácie je pochopiť problém pomocou konkrétnych prípadov daného problému. Vo chvíli, keď si žiak začne uvedomovať spoločné prvky (podobnosti riešenia) konkrétnych prípadov nastupuje stratégie zovšeobecnenia riešenia. Použitie uvedených dvoch stratégii demonštrujeme na úlohe kreslenia trojpruhovej vlajky (viď. nižšie).

Ak žiak nemá patričné skúsenosti s riešením problémov, prípadne je mu predložený veľmi náročný problém, môže mu učiteľ pomôcť pri hľadaní riešenia náročnejšej úlohy tým, že mu zadá vyriešiť vhodne pripravené jednoduchšie úlohy (konkretizácia). V príprave na programátorské a matematické súťaže sa zvyknú zostavovať súbory **prípravných (návodných) úloh**. Pre nadaných a nadšených žiakov môžeme vytvárať **rozširujúce úlohy**, ktoré sú náročnejšími verziami súťažných úloh s vyššou mierou zovšeobecnenia.

Našou snahou je, aby každý žiak bol schopný konkretizácie a následného zovšeobecnenia. U nadaných žiakov je cieľom dosiahnuť ich vnútornú potrebu (s minimálnou intervenciou učiteľa) ďalšieho skúmania úlohy prostredníctvom jej vyššej miery zovšeobecnenia (čo je typické pre vedeckú prácu).

Na úlohe s vykreslením trojpruhovej vlajky, ktorá je modifikovanou verziou úlohy zo súťaže PALMA junior (GUNIŠ – ŠNAJDER, 2013) ukážeme zadania jej prípravných úloh a tiež rozširujúcich úloh (ŠNAJDER – GUNIŠ – GUNIŠOVÁ, 2008).

- Pôvodná úloha:
    - *Vytvorte program, ktorý vykreslí vlajku s danou šírkou a výškou. Vlajka pozostáva z troch rovnako širokých farebných pruhov usporiadaných vodorovne alebo zvislo. (**vlajka3**)*
- Prípravné úlohy:
    - *Vytvorte program, ktorý vykreslí štvorec s konkrétnou dĺžkou strany. (**štvorec**)*
    - *Vytvorte program, ktorý vykreslí obdĺžnik s konkrétnymi dĺžkami strán. (**obdlžník1**)*
    - *Vytvorte program, ktorý vykreslí obdĺžnik s konkrétnymi dĺžkami strán vyfarbený konkrétnou farbou. (**obdlžník2**)*
    - *Vytvorte program, ktorý vykreslí obdĺžnik so zadanými dĺžkami strán vyfarbený konkrétnou farbou. (**obdlžník3**)*
    - *Vytvorte program, ktorý vykreslí obdĺžnik so zadanými dĺžkami strán vyfarbený zadanou farbou. (**obdĺžnik**)*
    - *Vytvorte program, ktorý vykreslí vlajku so zadanými dĺžkami strán s troma zvislými pruhmi vyfarbenými troma zadanými farbami. (**vlajka1, vlajka2**)*
- Rozširujúce úlohy:
    - *Vytvorte program, ktorý vykreslí vlajku so zadanými dĺžkami strán s troma pruhmi so zadanými farbami, zadaným pomerom šírok pruhov a zadanou orientáciou pruhov. (**vlajka4**)*
    - *Vytvorte program, ktorý vykreslí vlajku so zadanými dĺžkami strán, zadanými farbami viacerých pruhov rovnakej šírky a zadanou orientáciou pruhov. (**vlajka5**)*
    - *Vytvorte program, ktorý vykreslí vlajku so zadanými dĺžkami strán, zadanými farbami viacerých pruhov rôznej šírky, zadaným pomerom šírok pruhov a zadanou orientáciou pruhov. (**vlajka6**)*
    - *Vytvorte program, ktorý vykreslí obdĺžnikovú vlajku pokrytú viacerými farebnými obdĺžnikmi so stranami rovnobežne orientovanými so stranami obdĺžnikovej vlajky. (**hyperobdĺžnik**)*



Obrázok 1 – Diagram znázorňujúci následnosť prípravných, súťažnej a rozširujúcich úloh.

Pri zostavovaní systémov úloh, resp. prípravných úloh pre programátorské súťaže je veľmi užitočné mať prístup k rôznym zbierkam úloh. Na webových stránkach súťaží, môžeme nájsť v ich archívoch alebo v ich databázach zadania, prípadne komentované riešenia súťažných úloh. Súťaž PALMA junior postupne zverejňuje archív úloh: http://di.ics.upjs.sk/palmaj/riesenia_komentare.htm. Súťaž iBobor umožňuje vyhľadávať a riešiť súťažné úlohy v prostredí Bobrovo http://bobrovo.ibobor.sk/ucitel/indexUcitel.php, podobne aj súťaž PALMA http://palma.strom.sk/. Za zaujímavú funkcionalitu on-line databáz súťažných úloh považujeme možnosť pre učiteľov a žiakov diskutovať o úlohách a ich riešeniach, napr. diskusia k súťažným úlohám Korešpondenčného seminára z programovania http://www.ksp.sk/forum/forumdisplay.php?fid=10.

Námety úloh a prieskumných otázok pre učiteľov:

- Akým spôsobom vytvárate k zadanej úlohe pomocné (prípravné, návodné) úlohy?
- Vytvorte pomocné (prípravné, návodné) úlohy k zadaniu úlohy: *Napíšte program, ktorý vykreslí vlajku Európskej únie.*
- Akým spôsobom vytvárate k zadanej úlohe rozširujúce úlohy?
- Vytvorte aspoň jednu rozširujúcu úlohu k zadaniu úlohy: *Napíšte program, ktorý vykreslí hodiny s ručičkami.*

3 ANALÝZA VYBRANÝCH ÚLOH SÚŤAŽE PALMA JUNIOR

Stručnú charakteristiku súťaže PALMA junior sme uviedli v kapitole Informatické súťaže. Sme presvedčení o tom, že programovanie nie je len o písaní programového kódu. Programovanie chápeme v širšom zmysle, ako efektívny nástroj na riešenie problémov. Preto sú úlohy v tejto súťaži zamerané na riešenie reálnych problémov. Paradoxne, tieto reálne problémy sú často situované do virtuálneho Korytnačkova a sú problémami jeho obyvateľov – korytnačiek. Takýto prístup nám umožňuje dotknúť sa reálnych problémov z nášho sveta, ktoré by boli pre žiakov často neriešiteľné. V Korytnačkove totiž môžu platiť iné pravidlá ako v našej spoločnosti. Často ide o zjednodušenie a idealizáciu. Aj prostredníctvom riešenia týchto problémov však vieme žiakom priblížiť základné koncepty a princípy vybraných oblastí informatiky. Výber problémov, ich zameranie a formulácie sú orientované na rozvoj programátorských zručností, algoritmického myslenia a matematických poznatkov. Okrem vyššie uvedeného úlohami sledujeme aj dosahovanie ďalších cieľov. Úlohy sú zamerané na rozvoj estetického vnímania, etických aspektov spoločnosti, tvorivosti a pod. Tieto „sekundárne“ ciele dosahujeme vhodným zasadením čisto informatického problému do reálneho kontextu.

Atribúty súťažných úloh

Počas ôsmych rokov trvania súťaže sme postupne nachádzali a zaraďovali do úloh vybrané prvky z programovania, algoritmizácie a matematiky. Ich stručný prehľad uvádzame v nasledujúcej tabuľke:

programovanie	algoritmy	matematika
korytnačia grafika operácie s premennými vyhodnocovanie výrazov použitie viacerých korytnačiek príkaz opakovania s podmienkou príkaz opakovania zadaný počet krát podprogramy bez parametrov podprogramy s parametrami podprogram s návratovou hodnotou rekurzia určovanie podmienok	jednoduchý sekvenčný algoritmus, algoritmy na postupnosti hodnôt, algoritmy usporadúvania algoritmy vyhľadávania korytnačka ako vykonávateľ algoritmu kreslenie tvarov s určitými vlastnosťami náhodnosť na vymedzenom intervale prechádzanie 2D	celočíselná aritmetika algebrické výrazy korytnačia geometria logické operátory postupnosti čísel práca s karteziánskym súradnicovým systémom prevody jednotiek Pytagorova veta vlastnosti rovinných a priestorových útvarov výpočty s uhlami

programovanie	algoritmy	matematika
podmienky, zložené podmienky (vnorené) podmienené príkazy (vnorené) príkazy opakovania jednoduché a štruktúrované údajové typy (zoznam)	štruktúrou/2D oblasťou testovacie podmienky tvorba výrazov rekurzívne algoritmy ...	percentá a časti celku ...

Súťaž PALMA junior nie je určená len žiakom. Ďalšou cieľovou skupinou sú učitelia, najmä učitelia informatiky včítane budúcich učiteľov informatiky. Každá súťažná úloha na seba „nabaľuje“ aj ďalšie informácie:

- **Zadanie úlohy** je formulované ako reálny problém. Pri pochopení a nájdení podstaty problému sa vyžaduje čítanie s porozumením.
- **Autorské riešenie** je ukážkové riešenie. Nie vždy je to „najlepšie“ riešenie. Autorské riešenie je vytvorené vždy s ohľadom na vedomosti cieľovej skupiny žiakov.
- **Autorský komentár** je návodom a zdôvodnením autorského riešenia. Jeho súčasťou sú aj metódy a postupy ako riešenie objaviť.
- **Komentár k žiackym riešeniam** je vyhodnotením chýb, ktorých sa súťažiaci dopustili resp. aj vyzdvihnutím zaujímavých, originálnych a neočakávaných riešení. Komentár je cenným najmä pre učiteľov, ktorí tak získajú čiastočnú predstavu o tom, čo môžu očakávať od svojich žiakov pri riešení tejto úlohy.

Niektoré z vyššie uvedených častí sme pre potreby tejto monografie preformulovali a rozšírili. V tejto monografii uvádzame aj ďalšie informácie k jednotlivým úlohám:

- **V informáciách o úlohe** zdôvodňujeme zaradenie úlohy, jej ciele a poznatky potrebné k jej vyriešeniu. Uvádzame aj stratégie, ktoré môžeme použiť pri hľadaní riešenia.
- **Námety na prípravné a rozširujúce úlohy** sú minizbierkou ďalších úloh, ktoré môže učiteľ so svojimi žiakmi riešiť. Tieto úlohy sú formulované ako „holé“ úlohy, niekde len námety a často im chýba reálny obal – kontext, situácia. Predpokladáme však, že tvorivý učiteľ si tento obal dokáže vytvoriť sám s ohľadom na svojich žiakov. Ďalšie námety úloh môže učiteľ nájsť v učebniciach programovania (BLAHO – KALAŠ, 2006) (VARGA – BLAHO – ZIMANOVÁ, 1999) (BELUŠOVÁ et al., 2002).

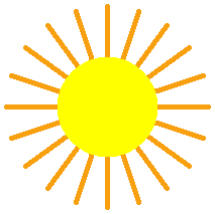
V nasledujúcej tabuľke uvádzame atribúty jednotlivých úloh uvedených v tejto monografii.

	programovanie	algoritmy	matematika	ostatné atribúty
Úloha o jesennom slniečku	príkazy korytnačej grafiky, príkaz opakovania	sekvenčný algoritmus so správnym poradím príkazov	výpočty s uhlami	stratégia rozkladu problému do podproblémov, estetika (výtvarná výchova)
Úloha o kreslení šachovnice	príkazy korytnačej grafiky, príkaz opakovania, podprogramy	prechod pravouhlou 2D oblasťou	celočíselná aritmetika	stratégia rozkladu problému do podproblémov, stratégia riešenia problému nájdi vzor
Úloha o tom, ako sa programuje vysávač	príkazy korytnačej grafiky, príkaz opakovania, opakovanie s podmienkou, podmienky	prechod 2D oblasťou	výpočty s uhlami, vlastnosti rovinných a priestorových útvarov (vzdialeností), logické operátory	časová zložitosť riešenia, nájdenie najvýhodnejšej trajektórie pohybu,
Úloha o súboji robotov	príkazy korytnačej grafiky, príkaz opakovania, podmienky, dátová štruktúra zoznam	algoritmus prechodu postupnosťou hodnôt (zadanými bodmi roviny)	logické operátory, práca s karteziánskym súradnicovým systémom	nájdenie najvýhodnejšej stratégie, zohľadnenie stratégie súpera, súboj algoritmov
Úloha o šetrnom vykurovaní	Aritmetické výrazy, vnorené podmienené príkazy	sekvenčný algoritmus	prevody jednotiek,	optimalizačná úloha (modelovanie javov z energetiky)
Úloha o treskúcej zime a snehovej vločke	príkazy korytnačej grafiky, príkaz opakovania, podmienky, rekurzia	kreslenie tvarov s určitými vlastnosťami, rekurzívne algoritmy	výpočty s uhlami	stratégia riešenia problému nájdi vzor
Úloha o prívesku z tvarovaného drôtika	dátová štruktúra zoznam, príkaz opakovania, využitie posúvača na zmenu hodnoty premennej	algoritmy na postupnosti hodnôt (morfing úsečiek)	algebrické výrazy, (parametrická rovnica úsečky)	stratégia rozkladu problému do podproblémov a použi vzorec, tvorba animácií
Úloha o smere vetra	príkaz vetvenia, resp. dátová štruktúra zoznam	sekvenčný algoritmus	Výpočty s uhlami, celočíselná aritmetika	stratégia rozkladu problému do podproblémov a odhadni – over – oprav, modelovanie javov z meteorológie
Úloha o trojskokanskej súťaži korytnačiek	podmienený príkaz	sekvenčný algoritmus	vlastnosti rovinných a priestorových útvarov	stratégia nakresli obrázok, nadbytočné dáta, šport
Úloha o tom, ako korytnačky v bludisku blúdili	príkazy korytnačej grafiky, cyklus s podmienkou, podmienené príkazy a podmienky	prechádzanie 2D oblasťou (prechod bludiskom,, analýza obrázka)	logické operátory	stratégia riešenia problémov – nakresli si obrázok (tabuľku)
Úloha o tom, ako Korypolícia kontroluje dodržiavanie dopravných predpisov	cyklus, podmienený príkaz, vnorené podmienky, logické výrazy, dátová štruktúra zoznam	algoritmy na postupnosti hodnôt (analýza dát, prechody medzi stavmi)	logické operátory	stratégia riešenia problému – vytvor tabuľku

Jednotlivé úlohy aj s doplňujúcimi informáciami uvádzame v nasledujúcej časti. Úlohy sú identifikovateľné pomocou číselníkov: „číslo ročníka – číslo kola – číslo úlohy“.

Úloha o jesennom slniečku

Informácie o úlohe



Úloha PJ 1-0-3 je zameraná na použitie problémovej stratégie rozkladu problému do podproblémov, príkazov korytnačej grafiky, príkazu opakovania s pevným počtom opakovaní, výpočtu veľkosti uhlov.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/3.doc

Vytvor procedúru `slniečko`, pomocou ktorej korytnačka vykreslí slniečko. V strede slniečka je žltý kruh, z ktorého vychádza 20 oranžových lúčov.

Svoje riešenie ulož ako Imagine projekt pod názvom **slniecko.imp**.

Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/riesenia_komentare/slniecko.IMP

Slniečko vytvoríme tak, že najprv vykreslíme slnečné lúče a potom slnečný kotúč. Lúče vykreslíme 20-krát opakovaným vykresľovaním úsečky z počiatočnej pozície dopredu a nazad a pootočením sa o uhol 360/20 stupňov.

```
viem slniečko
  ;lúče
  nechHrúbkaPera 5
  nechFarbaPera "oranžová
  opakuj 20 [
    dopredu 100
    vzad 100
    vpravo 360/20
  ]
  ;kotúč
  nechFarbaPera "žltá
  kruh 100
koniec
```

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/riesenia_komentare/uloha3.htm

Väčšina súťažiacich vyriešila úlohu správne. Súťažiaci okrem autorského riešenia objavili aj iné spôsoby kreslenia slniečka, napr. že lúče nevykresľovali od počiatku, ale od okraja kotúča. V nesprávnych riešeniach sa namiesto kruhového kotúča objavoval n-uholník, prípadne lúče nevychádzali z mysleného stredu kotúča. Len v zopár riešeniach sa namiesto uhla 18 stupňov objavoval výraz 360/20, ktorý je predpokladom pre zovšeobecnenie riešenia pre ľubovoľný počet lúčov. V jednom riešení bola namiesto počtu opakovaní 20 uvedená hodnota 360, čo si pri spustení

program jeho autor neuvedomil, že zbytočne veľakrát kreslí lúče. V niekoľkých riešeniach bol v programe uvedený príkaz `domov`, ktorý znemožňuje použitie procedúry pre umiestnenie slniečka na inú pozíciu. Zopár súťažiacich v riešeniach namiesto príkazu `kruh 100` použili menej štandardný postup – nastavenie hrúbky pera na 100 a posun o 0 krokov. Niekoľko začínajúcich súťažiacich poslali projekt s obrázkom slniečka bez programu.

Skúsenosti z letného tábora ukázali aj na inú typickú chybu pri riešení tejto úlohy, a to, že si žiaci neuvedomia v akom poradí majú vykresľovať lúče a kotúč. Je zaujímavé, že niektorým žiakov vôbec nevadí a nevnímajú ako chybu, že vykreslili lúče na už nakreslený kotúč, čo nie je v súlade so zadaním úlohy.

V riešení úlohy je možné namiesto príkazu `kruh 100` použiť tiež príkaz `bod 100`, ktorý ale neodporúčame používať, aby žiaci nenadobudli nesprávnu matematickú predstavu, že bod môže byť aj dvojrozmerným útvarom.

Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

Podstatou riešenia súťažnej úlohy je výpočet veľkosti uhlov (1), stratégia rozkladu problému do podproblémov (2), určenie správneho poradia príkazov (3), použitie príkazov korytnačej grafiky (4) a príkazu opakovania s pevným počtom opakovaní (5). Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

- Vykreslite trojlistú vrtuľku veterného mlyna. (1, 4)
- Na mieste činu zanechali postupne štyria ľudia prekrývajúce sa farebné kruhové stopy (červenú, zelenú, žltú, modrú). Vykreslite situáciu, ktorá ukazuje, že zlodejom je človek s červenou stopou. (3, 4)
- Vykreslite hrebeň s 20 zubami. (1, 2, 3, 4, 5)

Rozširujúce úlohy:

Súťažnú úlohu môžeme rozšíriť tak, že necháme žiakov dopracovať viac detailov, napr. vykresliť slniečko s očami a ústami. Podstatnejším rozšírením úloh rozšírením úlohy je použitie jedného a potom čo najväčšieho počtu parametrov v procedúre.

- Vykreslite slniečko s očami a ústami.
- Vykreslite slniečko so zadaným počtom lúčov.
- Vykreslite slniečko s veľkými a malými lúčmi so zadanou dĺžkou lúčov, hrúbkou pera, priemerom kotúča, farbou lúčov a kotúča.

Úloha o kreslení šachovnice

Informácie o úlohe

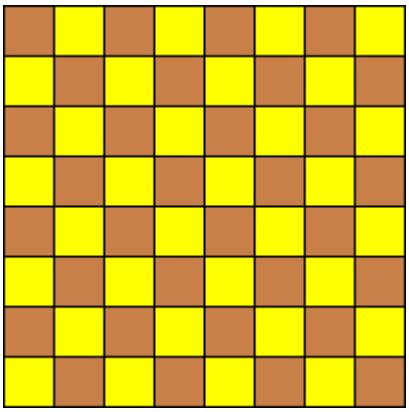
Úloha PJ 1-0-4 je zameraná na využitie stratégií riešenia problémov: rozklad problém na podproblémy a nájdenie vzoru. V riešení úlohy sa využívajú cykly, vlastnosti deliteľnosti čísiel a korytnačia grafika.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/4.doc

Vytvor procedúru `šachovnica`, pomocou ktorej korytnačka vykreslí šachovnicu 8×8. V šachovnici sa striedajú políčka žltej a hnedej farby.

Svoje riešenie ulož ako Imagine projekt pod názvom **sachovnica.imp**.

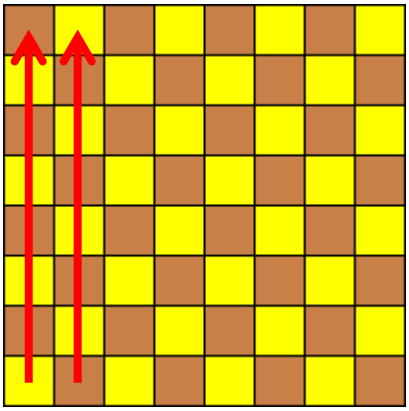


Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/riesenia_komentare/sachovnica1.IMP
http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/riesenia_komentare/sachovnica2.IMP
http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/riesenia_komentare/sachovnica3.IMP
http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/riesenia_komentare/sachovnica4.IMP
http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/riesenia_komentare/sachovnica5.IMP
http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/riesenia_komentare/sachovnica6.IMP

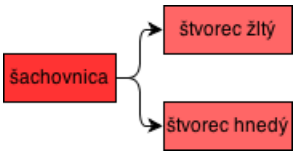
Túto úlohu možno riešiť viacerými spôsobmi, od prvoplánových až po premyslené riešenie využívajúce niektoré vlastnosti počítačovej grafiky a deliteľnosti čísiel.

Jednoduché riešenie spočíva v rozpoznaní a následnom vykreslení 64 štvorcov (v ôsmych radoch) striedajúcej sa farby do mriežky 8×8.



Obrázok 2 – Kreslenie šachovnice striedavým vykresľovaním farebných štvorčekov

Pri tomto riešení využijeme dve procedúry, procedúru na kreslenie žltého a hnedého štvorca. Hlavná procedúra premiestňuje kresliace pero (korytnačku) na vhodné pozície a striedavo volá procedúry na kreslenie farebných štvorcov.



Obrázok 3 – Dekompozícia problému kreslenia šachovnice na kreslenie štvorcov

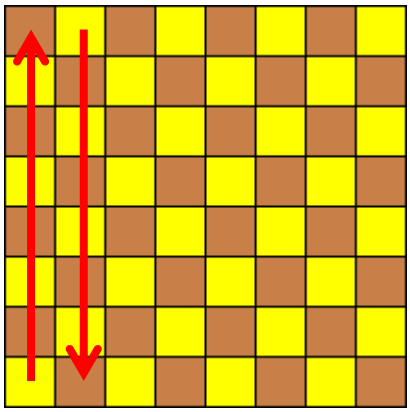
Nasledujúci kód by sa dal zjednodušiť využitím ďalších príkazov jazyka Imagine Logo. Našim zámerom však bolo ukázať riešenie tejto úlohy využitím čo najmenšieho počtu rôznych príkazov.

```
viem šachovnica
  ph
  opakuj 4 [
    opakuj 4 [
      štvorecŽltý
      do 25
      štvorecHnedý
      do 25
    ]
    vp 90
    do 25
    vp 90
    do 25*8
    vl 180

    opakuj 4 [
      štvorecHnedý
      do 25
      štvorecŽltý
      do 25
    ]
    vp 90
    do 25
    vp 90
    do 25*8
    vl 180
  ]
koniec
```

```
viem štvorecŽltý
  pd
  opakuj 4 [
    do 25
    vp 90
  ]
  nechFv "žltá
  vp 45
  ph
  do 10
  vyplň
  vz 10
  vl 45
koniec
```

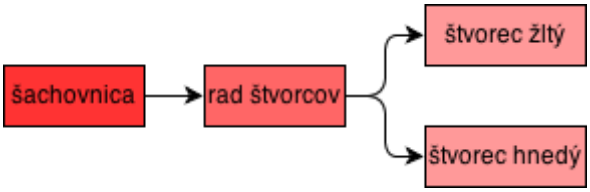
```
viem štvorecHnedý
  pd
  opakuj 4 [
    do 25
    vp 90
  ]
  nechFv "hnedá
  vp 45
  ph
  do 10
  vyplň
  vz 10
  vl 45
koniec
```

Pri ďalšom riešení si môžeme všimnúť, že v každom stĺpci sa opakuje rovnaká osmica štvorčekov. Striedavo ju však kreslíme zdola nahor a zhora nadol.



Obrázok 4 – Kreslenie šachovnice striedavým vykresľovaním stĺpcov

Pôvodnú schému rozdelenia problému na podproblémy sme nahradili schémou:



Obrázok 5 – Dekompozícia problému kreslenia šachovnice na kreslenie radov a štvorcov

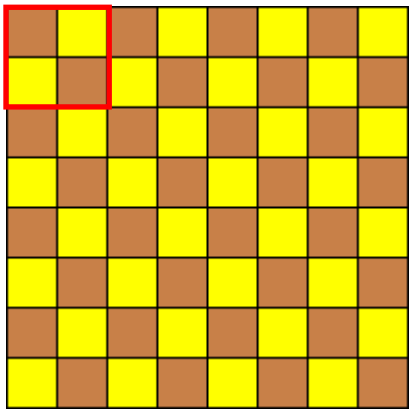
```
viem šachovnica
 ph
 opakuj 4 [
  radŠtvorcov
  vp 90
  do 50
  vp 90
  radŠtvorcov
  vl 180
 ]
koniec
```

```
viem radŠtvorcov
 opakuj 4 [
  štvorecŽltý
  do 25
  štvorecHnedý
  do 25
 ]
koniec
```

```
viem štvorecŽltý
 pd
 opakuj 4 [
  do 25
  vp 90
 ]
 nechFv "žltá
 vp 45
 ph
 do 10
 vyplň
 vz 10
 vl 45
koniec
```

```
viem štvorecHnedý
 pd
 opakuj 4 [
  do 25
  vp 90
 ]
 nechFv "hnedá
 vp 45
 ph
 do 10
 vyplň
 vz 10
 vl 45
koniec
```

Ďalšie, o niečo premyslenejšie riešenie využíva fakt, že okrem vzoru žltý a hnedý štvorec sa v obrázku opakuje aj ďalší útvar: „minišachovnica“ s rozmermi 2×2.



Obrázok 6 – Opakujúci sa vzor na šachovnici

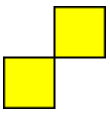
Tento prístup má aj tú výhodu, že netreba striedať smer kreslenia ako pri kreslení celých stĺpcov.

```
viem šachovnica
 ph
 opakuj 4 [
  opakuj 4 [

  miniŠachovnica
   do 2*25
  ]
  vz 4 * 50
  vp 90
  do 2 * 25
  vl 90
 ]
koniec
```

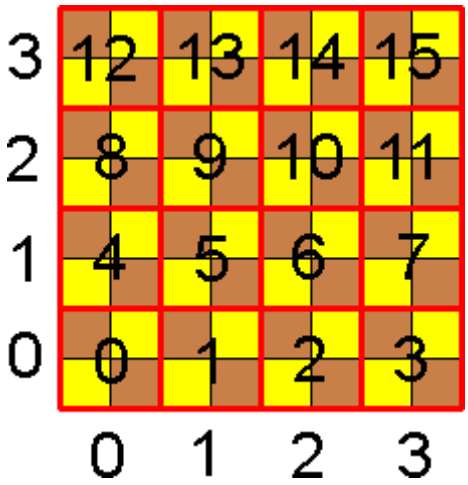
```
viem miniŠachovnica
 opakuj 2 [
  štvorecŽltý
  do 25
  štvorecHnedý
  do 25
  vp 90
  do 50
  vp 90
 ]
koniec
```

```
viem štvorecŽltý
 pd
 opakuj 4 [
  do 25
  vp 90
 ]
 nechFv "žltá
 vp 45
 ph
 do 10
 vyplň
 vz 10
 vl 45
koniec
```

```
viem štvorecHnedý
 pd
 opakuj 4 [
  do 25
  vp 90
 ]
 nechFv "hnedá
 vp 45
 ph
 do 10
 vyplň
 vz 10
 vl 45
koniec
```

V autorských riešeniach nájdeme aj ďalšie spôsoby kreslenia šachovnice (cik-cak, špirála a pod.) Za zmienku stojí posledné z autorských riešení využívajúce kreslenie štvorca 2×2, vlastnosti prekresľovania útvarov v počítačovej grafike a vlastnosti deliteľnosti čísiel. Pri tomto prístupe nakreslíme veľký hnedý štvorec, do ktorého budeme kresliť nasledujúcu vzorku  .

Pozície vzoriek objasňuje nasledujúci obrázok.



Obrázok 7 – Očíslovanie vzoriek v šachovnici

Všimnime si, že ak vhodne očíslujeme riadky, stĺpce a vzorky, tak z čísla vzorky vieme vypočítať číslo stĺpca (zvyšok po delení číslom 4) a číslo jej riadku (celočíselné delenie číslom 4). Kreslenie vzoriek sme si uľahčili využitím príkazu `polygón`.

```
viem šachovnica
 ph
 nechfv "hnedá
 polygón [ opakuj 3 [do 200 vp 90]]
 nechfv "žltá
 opakuj 16 [
   urobTu "riadok cPodiel (počítadlo-1) 4
   urobTu "stĺpec zvyšok (počítadlo-1) 4
   nechXYSúr 50 * :stĺpec 50 * :riadok
   polygón [do 25 vp 90 do 50 vl 90 do 25 vl 90 do 25 vl 90 do 50]
 ]
koniec
```

Toto riešenie využíva poznatky žiakov z matematiky (deliteľnosť čísiel), ktoré nám umožnia túto závislosť objaviť a použiť ju v riešení.

Táto úloha nám ukazuje množstvo rôznych prístupov vedúcich k riešeniu toho istého problému. Od žiakov však vyžaduje chuť hľadať a skúmať na prvý pohľad neviditeľné vzťahy a neuspokojenie sa len s prostým nájdením nejakého riešenia. Hľadanie rôznych riešení tejto úlohy je dobrý spôsob ako u žiakov rozvíjať logické, matematické a algoritmické myslenie.

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2005_2006/0/riesenia_komentare/uloha4.htm

Častou chybou v žiackych riešeniach bola presnosť kreslenia (hrubé čiary, kreslenie nie štvorcov) a efektívnosť kódu. Napriek tomu, že cyklus opakuj (cyklus s pevným počtom opakovaní) značne sprehľadní kód, žiaci sa mu často vyhli viacnásobným kopírovaním častí kódu.

Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

Úloha je zameraná na použitie stratégii: rozklad problém na podproblémy (1) a nájdenie vzoru (2), použitie príkazu opakovania (3), korytnačej grafiky (4) a z časti aj na matematické operácie zvyšok a podiel pri celočíselnom delení (5). Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

- Vykreslite štvorec, resp. štvorce v rôznych pozíciách, napr. za sebou, vedľa seba, pootočené. (1, 3, 4)

- Vykreslite štvorcovú mriežku, obdĺžnikovú mriežku. (1, 2, 3, 4)
- Vykreslite postupnosť tvarov skladajúcu sa z dvoch/troch tvarov, ktoré sa pravidelne striedajú. (1, 3, 4, 5)

Rozširujúce úlohy:

Súťažnú úlohu môžeme rozšíriť tak, že ju zadáme viac parametricky (napr. voliteľný rozmer šachovnice, obdĺžniková šachovnica). Ďalšou možnosťou je vypĺňanie plochy nie štvorcami, ale napr. trojuholníkmi. Výsledkom tak bude pyramída z rôznofarebných trojuholníkov. Pri ďalších úlohách môžeme požadovať vypĺňanie plochy viacerými útvarmi súčasne.

- Nakreslite šachovnic:
  - s rozmermi n×n.
  - s rozmermi m×n.

- Vykreslite šachovnicu tak, že do veľkého hnedého štvorca/obdĺžnika budete vykresľovať menšie žlté štvorčeky. Ich pozíciu určite podobne ako v poslednom riešení súťažnej úlohy.

- Vykreslite pyramídu pozostávajúcej z farebných trojuholníkov (2 farby, 4 farby).
- Nakreslite pohľad na kocku, ktorá je zložená z malých kociek striedajúcej sa farby.

Úloha o tom, ako sa programuje vysávač

Informácie o úlohe

Úloha PJ 3-4-3 je zameraná na vytvorenie efektívneho programu pre vykonávateľa (vysávač), testovanie podmienok a cyklus s podmienkou. Efektívnosť algoritmu je pre žiakov často nepodstatnou vlastnosťou algoritmov. Pri algoritmoch, ktoré žiaci navrhujú, je rozdiel niekoľkých tisícin až stotín sekúnd vo vykonávaní algoritmu nepodstatný. Len ťažko sa hľadajú problémy (na úrovni žiakov), kde by tento rozdiel bol významnejší a pre žiakov relevantný. Aj preto sme do súťaže zaradili úlohy, kde požiadavka na efektívnosť je jedným z hlavných kritérií hodnotenia algoritmu. Jednou z nich je úloha o programovaní robotického vysávača.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2007_2008/4/3.doc

Korytnačie mamy sú nadšené. V továrni na výrobu vysávačov začali vyrábať vysávače, ktoré by mali vedieť sami povysávať miestnosť. Mali by, ale nevedia. Problém je v tom, že v Korytnačkove je nedostatok programátorov. Vysávače sú totiž programovateľné.

Konštruktéri to však vymysleli celkom šikovne. Vysávač sa vie pohybovať smerom dopredu. Slúži na to príkaz `vpred počet_krokov`. Ak narazí do steny miestnosti, aktivuje sa niektorý zo senzorov a vysávač sa zastaví. Senzory sú štyri: `s1`, `s2`, `s3` a `s4` a sú umiestnené vpredu. Ak je senzor aktivovaný (môže ich byť súčasne aktivovaných viac), jeho hodnotou je `"áno`, inak má hodnotu `"nie`. Šírka vysávača je 31. Počet krokov, ktoré vysávač prejde, je uložený v premennej `"trasa`. Jej hodnotu môžete kedykoľvek vynulovať príkazom `nuluj`.

Korytnačie izby majú tvar osemuholníka, pričom všetky susedné steny zvierajú rovnaké uhly. Ak vysávač prejde po podlahe, zanechá stopu. Podľa toho vieme, ktoré časti izby sú už povysávané a ktoré ešte nie.

Tvojou úlohou je naprogramovať procedúru `vysávaj` tak, aby vysávač povysával celú izbu. Musíš ale šetriť energiou. Ak by vysávač povysával celú izbu bez toho, aby prešiel cez nejaké miesto viackrát, spotreboval by 100 % energie. To sa Ti asi nepodarí. Navrhni taký postup, aby bola spotreba energie čo najmenšia.

Stiahni si Imagine projekt **vysavac.imp**. Tvojou úlohou je doplniť procedúru `vysávaj`.

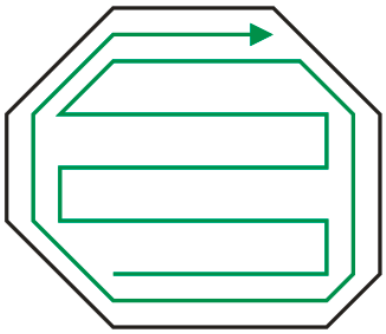
Svoje riešenie ulož ako Imagine projekt pod názvom **vysavac.imp**.

Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2007_2008/4/riesenia_komentare/vysavac.imp

Súčasťou zadania je pripravené prostredie, ktoré obsahuje procedúru na generovanie rôznych miestností a robota (vysávač) s obmedzenou funkcionalitou. Prvým problémom je nájsť trajektóriu pohybu robota tak, aby čo najmenej prechádzal opakovane po rovnakých častiach miestnosti. Druhým problémom je naprogramovať robota tak, aby prechádzal po nájdenej trajektórii. Využijeme ešte fakt, že vysávač má tvar osemuholníka, ktorého uhly strán sú zhodné s uhlami strán miestnosti.

Jedna z možných trajektórií je znázornená na obrázku:



Obrázok 8 – trajektória pohybu vysávača

Pohyb vysávača by sa dal rozpísať do nasledovných krokov:

1. Inicializácia, nastavenie sa do štartovacej pozície v rohu miestnosti,
2. Vysávanie vnútra, opakuj:
    1. pohyb rovno, pri náraze otočka vľavo,
    2. pohyb rovno, pri náraze otočka vpravo
3. Vysávanie obvodu miestnosti

Vysávač sa vždy nachádza na „spodnej“ strane miestnosti otočený smerom od spodnej steny miestnosti. Presunieme ho do rohu miestnosti a začneme vysávať vnútro miestnosti. Podľa toho, ktoré senzory sú aktivované (a ktorým smerov sa vysávač pohybuje) vieme usúdiť, v akej pozícii sa vysávač voči stenám miestnosti nachádza a podľa toho vieme korigovať jeho ďalší postup. Po každom náraze vysávač posunieme popri stene o jeho šírku (31 na vysávanie ďalšieho pásu miestnosti). Ak sa vysávač nedá posunúť o 31 bodov do ďalšieho radu vieme, že celé vnútro je povysávané a môžeme začať vysávať obvod miestnosti.

Výsledný program môže vyzerať nasledovne:

```
viem vysávaj
  urobTu "povysávané "nie
  urobTu "povysávanéVnútro "nie
```

```
vp 90
vpred 1000

kým [nieJe :povysávané] [
  ak nieJe :povysávanéVnútro [
    ak2 zároveň :s3 :s4 [
      nuluj vl 45 vpred 31
      ak2 :trasa < 31 [
        vl 45 vpred 31-:trasa vl 90
      ][
        vl 135
      ]
    ][
      ak2 zároveň :s2 :s3 [
        nuluj vl 90 vpred 31
        ak2 :trasa < 31 [
          vl 45 vpred 31-:trasa vl 45
        ][
          vl 90
        ]
      ][
        nuluj vl 135 vpred 31
        ak2 :trasa = 0 [
          urobTu "povysávanéVnútro "áno
        ][
          vl 45
        ]
      ]
    ]
    ak2 :povysávanéVnútro [
      vp 180
      opakuj 7 [
        vpred 1000 vp 45
      ]
      urobTu "povysávané "áno
    ][
      vpred 1000
    ]
  ]

  ak nieJe :povysávanéVnútro [
    ak2 zároveň :s1 :s2 [
      nuluj vp 45 vpred 31
      ak2 :trasa < 31 [
        vp 45 vpred 31-:trasa vp 90
      ][
        vp 135
      ]
    ][
      ak2 zároveň :s2 :s3 [
        nuluj vp 90 vpred 31
        ak2 :trasa < 31 [
          vp 45 vpred 31-:trasa vp 45
        ][
          vp 90
        ]
      ][
        nuluj vp 135 vpred 31
        ak2 :trasa = 0 [
          urobTu "povysávanéVnútro "áno
        ][
```

```
            vp 45
          ]
        ]
      ]
      ak2 zároveň :povysávanéVnútro nieJe :povysávané [
        vp 180
        opakuj 7 [
          vpred 1000 vl 45
        ]
        urobTu "povysávané "áno
      ][
        vpred 1000
      ]
    ]
  ]
koniec
```

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2007_2008/4/riesenia_komentare/3.htm

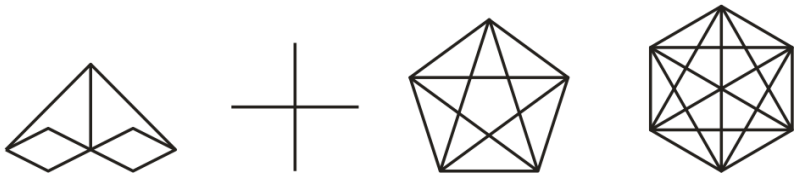
Zrejme najťažšou časťou úlohy bolo nájdenie trajektórie pohybu vysávača. Vysávač by mal povysávať celú miestnosť a to čo najefektívnejšie. Niektorí súťažiaci zvolili stratégiu pohybu po špirále z približného stredu miestnosti. Nápad to nie je zlý, ale ak je miestnosť pretiahnutá, tak popri stenách na dlhšej strane prejde vysávač zbytočne veľa krát. Navyše nevieme odhadnúť koľko krát sa má vysávač otočiť. Pri opačnej stratégii, špirále smerom dovnútra, sa stalo, že v strede ostala časť miestnosti nepovysávaná.

Námety na prípravné a rozširujúce úlohy

Súťažná úloha je zameraná na efektívnosť nájdeného riešenia (1), testovanie podmienok (2), cykly (3) a korytnačiu grafiku (4). Na precvičenie pojmu efektívnosť môžeme zradiť úlohy typu „nakresliť jedným ťahom“. Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

Prípravné úlohy:

- Nasledovné obrázky nakreslite jedným ťahom:
  - trojuholník, štvorec, n-uholník, (1, 3, 4)
  - hviezdu s párnym alebo nepárnym počtom cípov (1, 2, 3, 4)
- Pokúste sa nakresliť nasledovné obrázky jedným ťahom. Ak to nie je možné, svoj názor zdôvodnite a vykreslite ich tak, aby ste čo najmenej prekresľovali už nakreslené čiary. Diskutujte o tom, ktoré riešenie je najlepšie a prečo. (1, 2, 3, 4)



- Vytvorte program, ktorý zistí, ktorá z dvoch zadaných hodnôt je najmenšia. (2)
- Vytvorte program, ktorý zistí, ktorá z troch zadaných hodnôt je najmenšia. (1, 2)
- Vytvorte program, ktorý vypíše tri zadané hodnoty od najmenšej po najväčšiu. (1, 2)
- Naprogramujte vysávač tak, aby čo najefektívnejšie povysával štvorcovú (obdĺžnikovú) miestnosť známej/neznámej veľkosti. (1, 2, 3, 4)

Rozširujúce úlohy:

Súťažnú úlohu môžeme rozšíriť tak, že rozšírime funkcionalitu, resp. schopnosti vysávača. Náročnejšiu verziu úlohu vytvoríme aj tak, že pripustíme aj iné tvary miestnosti, chodby a pod.

- Naprogramujte nový vysávač podľa zadania súťažnej úlohy. Nový vysávač sa od pôvodného líši v tom, že
    - má senzor pre už navštívené miesto (vie zistiť, či už na danom mieste bol alebo nie),
    - má pamäť, do ktorej si môžeme ukladať pomocné dáta, napr. pozíciu, aktuálneho stavu a pod.
- Naprogramujte vysávač tak, aby vedel čo najefektívnejšie povysávať miestnosti rôznych tvarov (konvexné, nekonvexné).
- Naprogramujte vysávač tak, aby vedel čo najefektívnejšie povysávať sieť chodieb. Úlohu si môžete na začiatku zjednodušiť tak, že každou chodbou stačí prejsť len raz. Neskôr uvažujte tak, že treba povysávať celú plochu každej chodby.

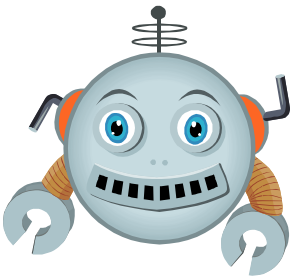
Úloha o súboji robotov

Informácie o úlohe

Úloha PJ 7-4-3 je zameraná na nájdenie stratégie zberu potravy. Táto úloha je však zaujímavá aj tým, že môžeme proti sebe postaviť dve stratégie a otestovať, ktorá z nich je lepšia. Takže pri hľadaní stratégie sa oplatí uvažovať aj o tom, ako „asi“ postupuje súper a podľa toho vylepšiť vlastnú stratégiu.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2011_2012/4/3.doc



O korytnačkách je známe, že sa nerady hýbu. A ak vôbec, tak veľmi pomaly. Viete si predstaviť ako dlho im trvá, kým vyzbierajú úrodu zeleniny na poli? Rozhodli sa preto, že na to použijú robota. Robota prinesú na pole a nechajú ho, nech pozbiera zeleninu. Problém je v tom, že roboty, ktoré sa v Korytnačkove predávajú, toho vedia veľmi málo. Poznajú súradnice každej sadenice zeleniny na poli, súradnice súpera a vedia vypočítať vzdialenosť dvoch bodov so zadanými súradnicami. A majú ešte automatické schopnosti, automaticky kráčajú a automaticky zoberú zeleninu, ku ktorej prišli. Je na každej korytnačke, aby si svojho robota naprogramovala podľa seba. Táto úloha spočíva v tom, otočiť robota správnym smerom k nejakej zelenine. Kráčať už bude sám. Keď vybranú zeleninu zoberie, treba ho natočiť k nejakej inej. To sa opakuje dovtedy, kým je ešte čo zbierať. Lenže, vyskytol sa ďalší problém. Na každé políčko zeleniny si robia nárok dve korytnačky. A každá má svojho robota. Robota teda treba naprogramovať tak, aby nielen zeleninu zozbieral, ale aby jej zozbieral viac ako robot druhej korytnačky.

Tvoja úloha je nasledovná. Vymysli a naprogramuj stratégiu pre robota korytnačky tak, aby bol čo najúspešnejší.

Stiahni si projekt **suboj.imp** a vytvor procedúru s `názvom svojho tímu`. Procedúra môže zmeniť len jednu vec, natočenie korytnačky podľa zvolenej stratégie. Ostatné sa deje automaticky.

Svoje riešenie ulož ako Imagine projekt **suboj.imp**.

Poznámka:

- *súradnice všetkých nezozbieraných sadeníc zeleniny sú uložené v premennej* `"súradnice`,
- *súradnice súperovho robota vráti procedúra* `pozSúper`,
- *vzdialenosť dvoch bodov so zadanými súradnicami vráti operácia* `vzdialenosť`,
- *názov stratégie pre každého z robotov napíš do príslušného textového políčka*,
- ***nevytváraj** žiadne iné pomocné procedúry a globálne premenné.*

Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2011_2012/4/riesenia_komentare/3_riesenie.imp

V nasledujúcom texte si postupne predstavíme niekoľko rôznych stratégií. Uvedené stratégie predstavujú aj spôsob, akým môžeme rozmýšľať a nachádzať lepšie stratégie.

Zrejme najprimitívnejšou stratégiou je čisto náhodný pohyb. I keď sa nám zdá byť táto stratégia prakticky nepoužiteľná, často ju používame, keď sme v strese a nie sme schopní sa rozhodnúť rozumnejšie (zablúdili sme v lese, hubár začiatočník hľadá huby a pod.)

```
viem stratégia0
  nechSmer ?
koniec
```

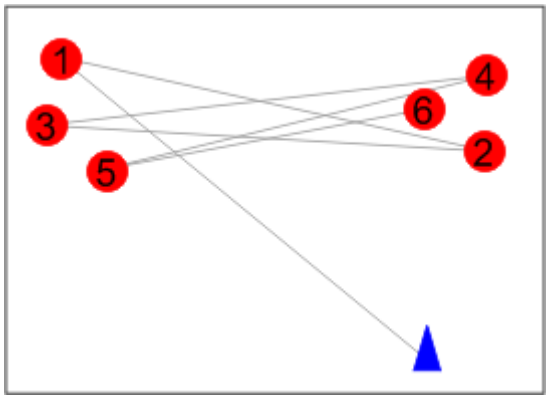
Tento prístup asi nie je najvhodnejší. Ak robot nájde nejakú zeleninu, je to len náhoda. Chcelo by to niečo premyslenejšie. Postupovať môžeme napr. tak ako začínajúci hubári, ktorí si uvedomili, že náhodné rozhodnutia vedú k úspechu len v mizivom počte prípadov. Keď hubári nevedia kde a ako hľadať, pôjdu za nejakým skúsenejším hubárom. Ak skúsený hubár natrafí na dobré miesto, bude tam prvý. Je ale šanca, že sa nám z jeho nálezu tiež niečo ujde. Nechajme teda robota v ďalšej stratégii, aby zamieril k súperovi.

```
viem stratégia1
  nechSmer smerK pozSúper
koniec
```

Je to lepšia stratégia, ale ak by súper použil stratégiu 0 alebo 1, asi by sme sa ďaleko nedostali. Trochu lepší prístup (pravdupovediac o dosť lepší) je, zbierať zeleninu v tom poradí, ako je uvedená v zozname súradníc. Toto je už naša vlastná stratégia a nespoliehame sa na šikovnosť súpera.

```
viem stratégia2
  nechSmer smerK prvý :súradnice
koniec
```

Už sme použili nejaký systematický prístup, pohyb robota nie je náhodný ale riadi sa jasnými pravidlami. Problém je v tom, že poradie sadeníc zeleniny v zozname nemusí byť najvhodnejšie. Pozrime si nasledujúci obrázok.

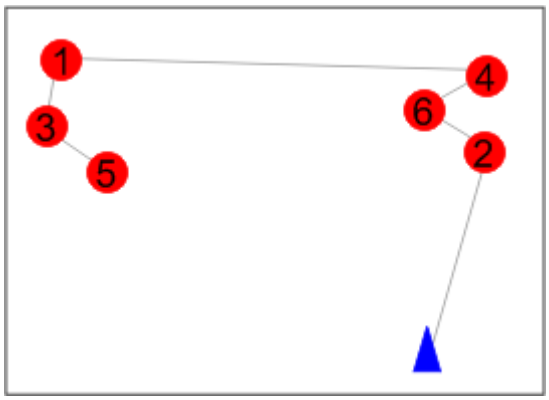


Obrázok 9 – Stratégia zberu potravy: podľa poradia v zozname

Robot zbytočne pobehuje z jednej strany na druhú, namiesto toho, aby najskôr vyzbieral jednu stranu a potom druhú. Vylepšime teda stratégiu tak, aby robot vždy šiel k najbližšej potrave.

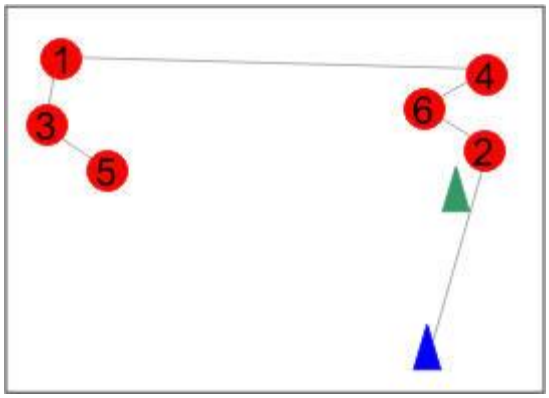
```
viem stratégia3
  urobTu "vzdialenosť vzdialenosť poz prvý :súradnice
  urobTu "kam prvý :súradnice
  prePrvky "bod :súradnice [
    ak vzdialenosť poz :bod < :vzdialenosť [
      urobTu "kam :bod
      urobTu "vzdialenosť vzdialenosť poz :bod
    ]
  ]
  nechSmer smerK :kam
koniec
```

Prejdeme zoznamom súradníc sadeníc a pre každú sadenicu si vypočítame našu vzdialenosť od nej. Ak nájdeme nejakú bližšiu ako sme doteraz našli, zapamätáme si ju. Takto docielime to, že si nakoniec pamätáme tú, ku ktorej sme najbližšie. Robota natočíme priamo k nej.



Obrázok 10 – Stratégia zberu potravy: podľa najbližšieho

Toto už vyzerá ako celkom dobrá stratégia. Nepočítame však s tým, že na poli je aj druhý robot.



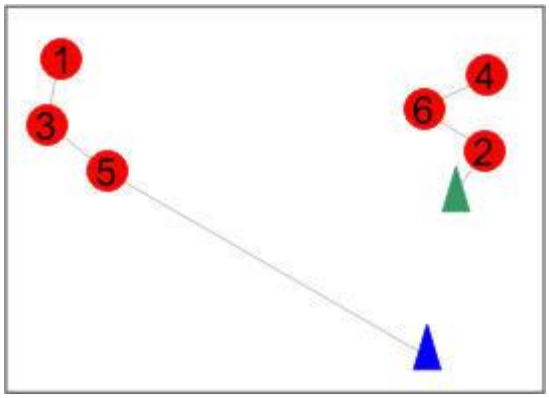
Obrázok 11 – Stratégia zberu potravy: podľa najbližšieho verzus druhý hráč

Kým sa modrý robot dostane k 2, tak zelený robot (ak použije rovnakú stratégiu) stihne pozbierať zeleninu 2, 4 a 6. V tejto situácií je pre modrého jasné, že sa mu neoplatí ísť do pravej časti, ale rovno zamieriť vľavo. Takže ísť k najbližšej sadenici sa oplatí len vtedy, ak tam súper nebude skôr.

Nasledujúca stratégia by teda mohla byť takáto. Nájdeme si najbližšiu sadenicu a vzdialenosť od nej. Potom si vypočítame, po koľkých krokoch príde k našej sadenici súperiaci robot (pozor, nemusí k nej ísť priamo). Ak tam bude skôr ako náš robot, nemá zmysel tam ísť. Vyberieme si druhú najbližšiu a opäť zistíme, po koľkých krokoch k nej dorazí súperiaci robot. Ak by mu to trvalo dlhšie, zamierime k nej. V opačnom prípade skúsime tretiu najbližšiu sadenicu. Takto pokračujeme dovtedy, kým nenájdeme sadenicu, ku ktorej sa náš robot dostane skôr ako súper. Ak taká neexistuje, nemá náš robot šancu vôbec nejakú sadenicu zobrať (ak samozrejme súperiaci robot používa stratégiu najbližšej sadenice).

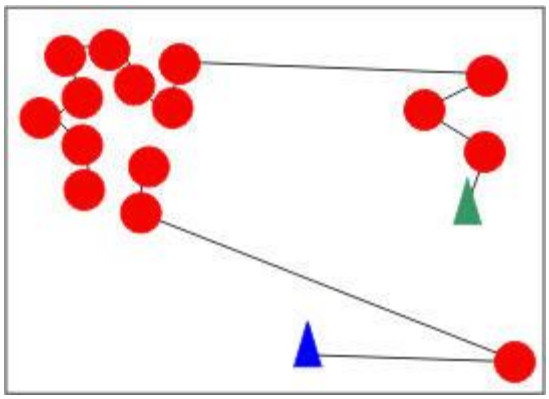
```
viem stratégia4
  urobTu "cieľ "nie
  urobTu "body :súradnice

  kým [nieJe :cieľ] [
    urobTu "vzdialenosť vzdialenosť poz prvý :body
    urobTu "kam prvý :body
    prePrvky "bod :body [
      ak vzdialenosť poz :bod < :vzdialenosť [
        urobTu "kam :bod
        urobTu "vzdialenosť vzdialenosť poz :bod
      ]
    ]

    urobTu "dĺžkaCestySúpera 0
    urobTu "odkiaľSúper pozSúper
    urobTu "bodySúpera :body
    urobTu "kamSúper []

    kým [zároveň :dĺžkaCestySúpera<:vzdialenosť :kamSúper<>:kam][
      urobTu "vzdialenosťSúpera vzdialenosť :odkiaľSúper prvý :bodySúpera
      urobTu "kamSúper prvý :bodySúpera
      prePrvky "bod :bodySúpera [
        ak vzdialenosť :odkiaľSúper :bod < :vzdialenosťSúpera [
          urobTu "kamSúper :bod
```

```
          urobTu "vzdialenosťSúpera vzdialenosť :odkiaľSúper :bod
        ]
      ]
      urobTu "dĺžkaCestySúpera :dĺžkaCestySúpera + :vzdialenosťSúpera
      urobTu "odkiaľSúper :kamSúper
      urobTu "bodySúpera bezVýskytov :kamSúper :bodySúpera
    ]

    ak :dĺžkaCestySúpera >= :vzdialenosť [
      urobTu "cieľ "áno
    ]

    ak zároveň :dĺžkaCestySúpera<:vzdialenosť :kamSúper=:kam [
      urobTu "body bezVýskytov :kam :body
      ak prázdny? :body [
        urobTu "cieľ "áno
      ]
    ]
  ]
  nechSmer smerK :kam
koniec
```



Obrázok 12 – Stratégia zberu potravy: najbližšia, ktorú dosiahnem skôr ako súper

Toto je už pomerne dobrá stratégia. Ale, existuje aj lepšia. Problémom tejto stratégie je to, že robot sa orientuje len na svoje najbližšie okolie. Nepozerá sa na rozmiestnenie sadeníc globálne. Zamieri teda k blízkej, osamotenej sadenici namiesto toho, aby zamieril k vzdialenejšiemu zhluku sadeníc. Túto situáciu ukazuje nasledujúci obrázok.



Obrázok 13 – Stratégia zberu potravy: najbližšia, ktorú dosiahnem skôr ako súper verzus vzdialenejší veľký zhluk

Robot by teda mal uvažovať nie len o sadeniciach ako takých, ale aj o skupinkách sadeníc. Navštíviť väčšiu, i keď vzdialenejšiu skupinku môže byť výhodnejšie než bližšiu a menšiu skupinku. Zároveň však musí predpokladať, ako bude postupovať súperiaci robot, aby sa nestalo, že kým dorazí k veľkej skupine sadeníc, súperiaci robot ju vyzbiera. Naprogramovať túto stratégiu už vôbec nie je jednoduché. Navyše vyhodnotenie najvhodnejšieho smeru by bolo zrejme tak náročné, že by zabralo príliš veľa času.

Poznámka: Podobný problém (i keď bez súperiaceho robota) riešia informatici už pomerne dlho a otázkou je, či vôbec existuje nejaké šikovné riešenie. Tento problém sa nazýva "Problém obchodného cestujúceho". Cieľom obchodného cestujúceho je vybrať sa z mesta, v ktorom sa práve nachádza, navštíviť každé mesto práve raz a vrátiť sa do počiatočného mesta. Pritom sa snaží túto cestu absolvovať tak, aby precestoval čo najmenšiu vzdialenosť, zaplatil čo najmenej peňazí a podobne. Viac sa o ňom dozviete napr. tu: http://sk.wikipedia.org/wiki/Probl%C3%A9m_obchodn%C3%A9ho_cestuj%C3%BAceho.

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2011_2012/4/riesenia_komentare/uloha3.htm

Viac ako polovica súťažiacich objavila stratégiu najbližšej sadenice. Táto však nijako nezohľadňuje súperiaceho robota a robot postupuje nezávisle od súperiaceho robota.

Objavil sa aj pokus využiť ťažisko sadeníc (i keď nie dotiahnutý do konca). Problémom tejto stratégie je, že v okolí ťažiska sa nemusí nachádzať žiadna sadenica (ak sú napr. sadenice rozmiestnené rovnomerne v protiľahlých rohoch poľa).

Niektorí použili vo svojom riešení nedovolené príkazy (napr. zmena pozície) alebo procedúre pridali vstupné parametre.

Poznámka: V súťaži PALMA junior sa pri vzájomnom porovnávaní stratégii súťažiacich umiestňovali roboti na náhodné pozície. To spôsobilo, že niektorá stratégia mohla byť umelo zvýhodnená lepšou štartovacou pozíciou. Odporúčame preto, aby aj voľba štartovacej pozície bola súčasťou vytvorenej stratégie. Tu však treba zohľadniť aj poradie v akom si stratégie volia štartovaciu pozíciu. Možným riešením je štartovaciu pozíciu voliť bez využitia pozície súperiaceho robota.

Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

Súťažná úloha bola zameraná na nájdenie vhodnej stratégie (1) prechodu určenými bodmi, sekundárne na prácu v súradnicovom systéme (2). Pri voľbe stratégie bolo potrebné zohľadniť

aj súperovu stratégiu, vďaka čomu sa podmienky úlohy počas jej riešenia menili. Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

- Nájdite najkratšiu trajektóriu pre vopred daný počet kusov potravy. (1, 2)
- Nájdite najkratšiu trajektóriu ak potravu zbiera len jeden robot. Svoju stratégiu postupne vylepšujte. (1, 2)

Rozširujúce úlohy:

Náročnejšie verzie súťažnej úlohy môžu spočívať vo voľbe premyslenejšej stratégie alebo v simulácii rôzneho správania sa robotov.

- Navrhnite stratégiu, ktorá zohľadňuje namiesto jednotlivých sadeníc ich zhluky. V tomto prípade je zaujímavé, ako charakterizovať zhluk a ako ho ohodnotiť.
- Navrhnite stratégiu tak, aby sa robot pri zbere potravy nepribližoval (ak je to možné) k súperovi.

Úloha o šetrnom vykurovaní

Informácie o úlohe

Úloha PJ 8-1-6 je reálne formulovaný problém, s ktorým sa môže stretnúť aj človek. Úlohou je nájsť model (matematický predpis) výpočtu efektívnosti kúrenia a vybrať (porovnávanie hodnôt) najvýhodnejšie palivo. V úlohe sa pracuje s reálnymi dátami i keď so zjednodušeným modelom. K tejto úlohe bolo vytvorené prostredie, ktoré obsahovalo formulár pre zadávania hodnôt jednotlivých parametrov.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2012_2013/1/6.doc

V Korytnačkove, podobne ako u nás, pomaly ale isto prichádza zima. A v zime býva zima zimúca. Korytnačky už teraz pobehujú po Korytnačkove a rozmýšľajú, radia sa ako mať v zime teplo a minúť pritom čo najmenej peňazí.

Rozhodnutie nie je vôbec jednoduché. Predajcovia paliva ponúkajú drevo, plyn a elektrickú energiu. Ich cena je rôzna a predajcovia ju uvádzajú za rôzne jednotkové množstvá (jm). Pri dreve sú to priestorové rovnané metre (prm), pri plyne a elektrine kilowatthodiny (kWh).

Z každého druhu paliva možno spaľovaním získať určité množstvo energie (megajoul – MJ). Množstvo energie, ktorú možno získať z jednotkového množstva daného paliva, uvádza predajca pri každom druhu paliva (MJ/jm). A nakoniec, každé palivo sa spaľuje v inom type kotla. Kotol, podľa toho ako je navrhnutý, prevedie na teplo v domácnosti len časť energie z paliva. Zvyšok sa nevyužije. Napr. pri spaľovaní dreva a plynu časť energie unikne do komína. Najmenej stratové sú elektrické kotle. Časť získanej energie, ktorú využije kotol na teplo v domácnosti, sa nazýva účinnosť kotla a meria sa v percentách.

Isto uznáte, že vyznať sa v tomto mori údajov nie je jednoduché. Aby to mali korytnačky aspoň trochu ľahšie, dohodli sa predajcovia kotlov, že všetky kotly budú ponúkať za rovnakú cenu. Ale aj tak je to pre korytnačky náročná úloha. Vedel by si im pomôcť?

Stiahni si projekt **palivo.imp** a doprogramuj procedúru `zistiNajvýhodnejšiePalivo`. Do zelených textových políčok si korytnačky zadajú údaje od predajcov. V červenom textovom políčku sa musí zobraziť najvýhodnejšie palivo. Ak sú niektoré palivá rovnako výhodné, je jedno ktoré z nich sa použije. Svoje riešenie ulož ako Imagine projekt **palivo.imp**.

Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2012_2013/1/riesenia_komentare/palivo_riesenie.imp

Riešenie tejto úlohy, aj napriek tomu, že pracujeme s množstvom dát, nie je až také zložité. Najskôr musíme zistiť, ako odmerať cenu vykurovania pre jednotlivé palivá. Potom nám najnižšia z nich určí najvýhodnejšie palivo.

Pozrime sa najskôr na cenu vykurovania. Faktory, ktoré na ňu vplývajú sú*:

- cena za jednotkové množstvo paliva,
- množstvo energie získané spálením jednotkového množstva paliva,
- účinnosť kotla, ktoré palivo spaľuje.

Do celkovej ceny vykurovania by sme mali zahrnúť aj cenu kotla (ako počiatočnú, jednorazovú investíciu). Keďže sú však všetky kotle rovnako drahé, bude mať ich cena pri každom palive rovnaký vplyv na cenu vykurovania. Pri porovnaní výhodností jednotlivých palív ju teda nemusíme brať do úvahy.

V konečnom dôsledku nás bude zaujímať pomer množstva ceny, ktorú sme zaplatili za palivo a energie, ktorú sme využili na kúrenie. Z celkovej energie, ktorú získame horením konkrétneho paliva, využijeme len časť. Akú veľkú to záleží od účinnosti kotla. Cenu za využitú energiu vypočítame nasledovne: $\text{cena za využitú energiu} = \frac{\text{cenaZaJednotkuPaliva}}{\text{výhrevnosťZJednotkyPaliva} \times \text{účinnosťKotla}}$

Napr. Predajcovia palivového dreva ponúkajú drevo za cenu 55 €/prm. Výhrevnosť palivového dreva je 11000 MJ/prm. Účinnosť kotla na drevené palivo je približne 60 %. To znamená, že cena za využitú energiu z horenia dreva je:

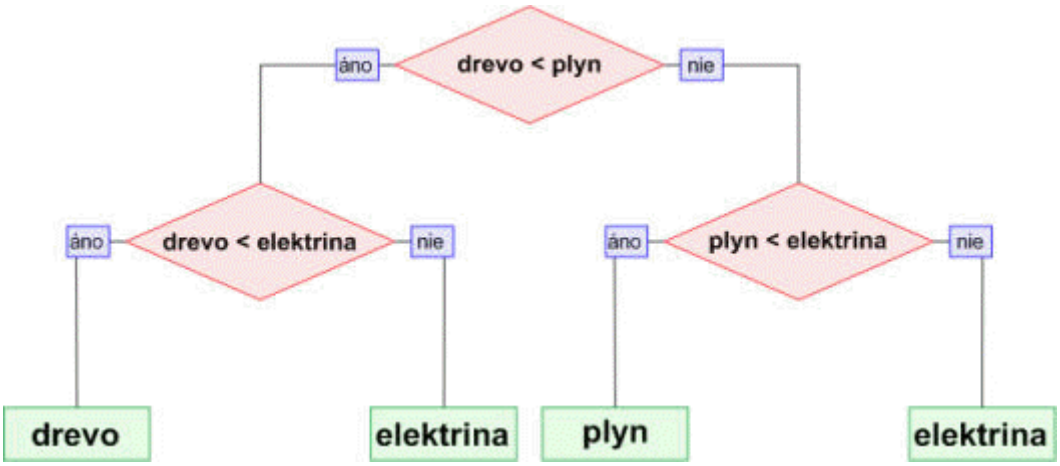
$$\text{cena za využitú energiu z dreva} = \frac{55\ €/\text{prm}}{11000\ \text{MJ/prm} * 60\ \%} = 0{,}0083\ €/\text{MJ}$$

Takto vieme vypočítať cenu vykurovania pre každý typ paliva. Najmenšia cena za vykurovanie nám určí najvýhodnejšie palivo.

Všetky potrebné dáta sú uvedené v textových políčkach pripraveného prostredia. Procedúra `zistiNajvýhodnejšiePalivo` by zatiaľ mohla vyzerať nasledovne:

```
viem zistiNajvýhodnejšiePalivo
  urobTu "drevo drevoCena / (drevoVýhrevnosť * drevoÚčinnosťKotla / 100)
  urobTu "plyn plynCena / (plynVýhrevnosť * plynÚčinnosťKotla / 100)
  urobTu "elektrina elektrinaCena / (elektrinaVýhrevnosť *
elektrinaÚčinnosťKotla / 100)
koniec
```

Ďalším krokom je zistiť, ktorá z premenných `"drevo`, `"plyn` a `"elektrina` má najmenšiu hodnotu. Musíme ich teda navzájom porovnať. Spravme to ale čo možno najšikovnejšie, na čo najmenší počet porovnaní. Naše skúsenosti z výučby ukazujú, že žiaci majú tendenciu vytvárať sériu zložených podmienok namiesto vnárania podmienok do seba. Pri návrhu testovacích podmienok nám môže pomôcť nasledujúci rozhodovací strom:



Obrázok 14 – Rozhodovací strom pre výber najvýhodnejšieho paliva

Ak prepíšeme tento rozhodovací strom do procedúry, dostaneme nasledujúci kód procedúry `zistiNajvýhodnejšiePalivo`:

```
viem zistiNajvýhodnejšiePalivo
  urobTu "drevo drevoCena / (drevoVýhrevnosť * drevoÚčinnosťKotla / 100)
  urobTu "plyn plynCena / (plynVýhrevnosť * plynÚčinnosťKotla / 100)
  urobTu "elektrina elektrinaCena / (elektrinaVýhrevnosť *
elektrinaÚčinnosťKotla / 100)

  ak2 :drevo < :plyn [
    ak2 :drevo < :elektrina [
      urobTu "naj "drevo
    ][
      urobTu "naj "elektrina
    ]
  ][
    ak2 :plyn < :elektrina [
      urobTu "naj "plyn
    ][
      urobTu "naj "elektrina
    ]
  ]
  najVýhodnejšiePalivo'nechHodnota :naj
koniec
```

(*) Takýto jednoduchý prístup platí len v Korytnačkove. V skutočnosti by sme do takejto ceny mali započítať aj ceny kotlov, ktoré sú rôzne a počiatočné náklady na montáž kotla. Ďalej by sme do úvahy mali brať aj množstvo kúpeného paliva, pretože predajca zvyčajne poskytuje aj nejakú množstvovú zľavu. Nezanedbateľnú položku tvoria aj náklady na prevádzku a údržbu kotla, ako napr. opravy, kontroly a pod.

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2012_2013/1/riesenia_komentare/uloha6.htm

Pri opravovaní tejto úlohy sme zistili zaujímavú vec. Väčšina zo súťažiacich zisťovala množstvo získanej energie za dané peniaze a nie cenu získaného množstvo energie. Z výsledných hodnôt potom vyberali tú maximálnu a nie minimum ako my v autorskom riešení.

Častou chybou bolo chybné porovnávanie hodnôt a určenie tej najvýhodnejšej. Pri úlohách tohto typu preto odporúčame žiakom si postup porovnávania ešte pred samotným programovaním zobraziť pomocou obrázka, resp. prehľadnej schémy.

Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

Súťažná úloha bola zameraná na vytvorenie modelu, ktorý by čo najlepšie reprezentoval reálnu situáciu (1), efektívne porovnanie hodnôt (2) a použitie formulárových objektov prostredia Imagine Logo (3). Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

- Vytvorte program, ktorý porovná dve/tri zadané hodnoty a vypíše najmenšiu/najväčšiu z nich. V procese porovnávania použite čo najmenší počet porovnaní.
- Vytvorte program pre výpočet ceny nákupu. Definujte si rôzne akcie predajcu, napr.: množstvová zľava, akciové ceny, kupóny, rôzne iné predajné akcie a ich možné kombinácie.

Rozširujúce úlohy:

Súťažná úloha bola formulovaná ako zjednodušený model reality. Náročnejšiu verziu úlohy získame, ak zohľadníme aj ďalšie reálne náklady súvisiace s vykurovaním.

- Vytvorte program pre zistenie najvýhodnejšieho paliva. Pri výpočte nákladov uvažujte aj:
  - rôzne ceny za kúpu kotla podľa typu spaľovania,
  - počiatočné investície (zavedenie plynovej prípojky, vybudovanie skladovacích priestorov na drevo a pod.),
  - cenu inštalácie kotla,
  - prevádzkové náklady,
  - životnosť kotla,
  - množstvo energie (napr. za rok) potrebnej na vykúrenie objektu, výkon kotla a pod.

Úloha o treskúcej zime a snehovej vločke

Informácie o úlohe

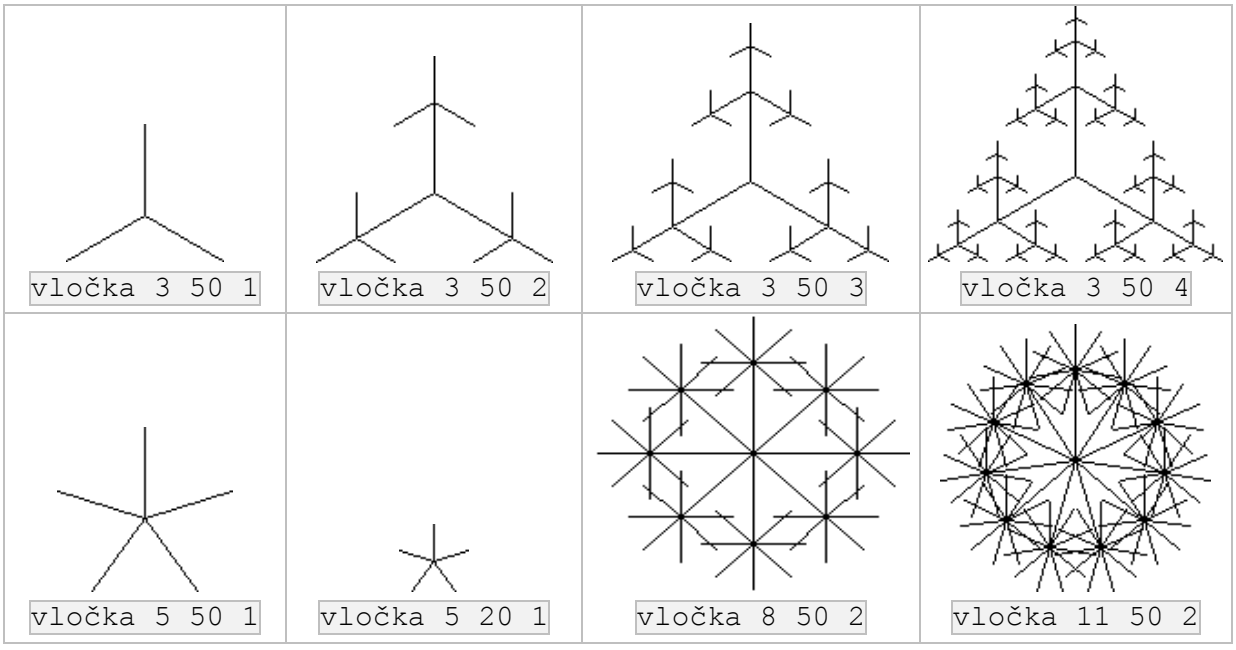
Úloha PJ 1-2-6 je úloha zameraná na pochopenie a použitie jednoduchej rekurzie. Predstavuje náročnejšiu verziu úlohy o kreslení binárneho stromčeka, s ktorou sa žiaci bežne stretnú pri výučbe tejto témy. Zatiaľ, čo pri kreslení binárneho stromčeka z každej vetvy vyrastajú dve menšie pod daným uhlom, v tomto prípade je počet vetvičiek voliteľný a uhol, ktorý zvierajú je odvodený od ich počtu. Analogicky k tomu je definovaná rekurzívna procedúra. Touto úlohou sme chceli otestovať v súťaži PALMA junior, do akej miery si žiaci osvojili rekurziu a či ju vedia použiť aj pri riešení „neučebnicových" úloh.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2005_2006/2/6.doc

Vonku je zima a poriadne prituhuje. Za oknom sa naháňajú snehové vločky. Všimli ste si, aké sú snehové vločky krásne? A každá je iná. Jedna má tri ramená, iná ich má päť, ďalšia ich má osem. Jedna má na konci ramienok ďalšiu, o polovicu menšiu vločku. A aj tá menšia vločka má na konci ramienok ďalšie, od nej o polovicu menšie vločky. Ale aj táto má na konci ramienok ďalšie vločky. Také malé, že ich už ani poriadne nevidno.

Tvoja úloha je jednoduchá. Vytvor procedúru `vločka`, pomocou ktorej korytnačka nakreslí snehovú vločku. Procedúre zadáme, koľko ramien má vločka mať, aké majú byť veľké a koľko rôznych malých vločiek je na konci jej ramien.



Svoje riešenie ulož ako Imagine projekt pod názvom **zima.imp**.

Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2005_2006/2/riesenia_a_komentare/vlocka.IMP

Rekurzívne úlohy sú veľmi náročné (ak sa neuspokojíme len s formálnym zapamätaním si nejakého algoritmu) na abstraktné myslenie, myslenie vo viacerých úrovniach a predstavivosť. Pri návrhu jednoduchých rekurzívnych algoritmov na kreslenie rekurzívnych obrázkov odporúčame postupovať nasledovne.

V prvom kroku by sme mali rozpoznať základný tvar, z ktorého sa výsledný obrázok skladá. V ukážkových vločkách sú to tie, ktorých počet „rôznych“ malých vločiek je práve 1.

Potom nájdeme postup, ako nakresliť jednu vločku so zadanou veľkosťou a zadaným počtom ramien. Nakreslíme jedno rameno, vrátime sa späť a otočíme sa. Toto opakujeme toľkokrát, koľko je ramien vločky. Veľkosť uhla otočenia závisí od počtu ramien. Prvá verzia procedúry kreslí vločku, ktorá má `početRamien` ramien a každé má dĺžku `:dĺžka`.

```
viem vločka :početRamien :dĺžka
  opakuj :početRamien [
    do :dĺžka
    vz :dĺžka
    vl 360 / :početRamien
  ]
koniec
```

Procedúra vločka však nakreslí len jednu, jednoduchú vločku. Na konci jej ramien nie je nič. V tejto etape sa musíme zamyslieť, kde v priebehu kreslenia veľkej vločky začneme kresliť vločku menšiu. Malo by to byť na konci jej ramien. Po nakreslení jedného ramena „prerušíme“ kreslenie vločky a spustíme kreslenie menšej vločky. Táto úvaha, prerušenie kreslenia veľkej vločky, rekurzívne vnáranie sa do kreslenia menších vločiek, vynorenie sa z rekurzie a pokračovanie v nedokončenom kreslení veľkej vločky je pomerne náročné na pochopenie. V tomto prípade je to ešte náročnejšie, pretože rekurzívne vnáranie sa deje opakovanie v cykle.

Ak chceme kresliť na konci jej ramien aj ďalšie, menšie vločky, upravíme ju nasledovne:

```
viem vločka :početRamien :dĺžka
  opakuj :početRamien [
    do :dĺžka
    vločka :početRamien :dĺžka / 2
    vz :dĺžka
    vl 360 / :početRamien
  ]
koniec
```

Táto procedúra by teoreticky nikdy neskončila. Kreslenie každej vločky by sme po nakreslení jedného jej ramena prerušili a spustili kreslenie ďalšej, menšej vločky. Musíme teda vymyslieť, ako kreslenie ďalších, menších vločiek zastaviť. Použijeme na to ešte jeden parameter – `:úroveň`, ktorý

bude číslovať, koľko úrovní vločiek je potrebné ešte vykresliť. Ak napr. najväčšia vločka obsahuje celkom 5 úrovní vločiek (včítane samej seba), vločka na konci jej ramena ich bude obsahovať len 4. Pri každom volaní parameter úroveň preto zmenšíme o 1. Ak má parameter `:úroveň` hodnotu aspoň 1, vieme, že potrebujeme vykresliť ešte aspoň jednu vločku. V opačnom prípade už nevykreslíme nič.

```
viem vločka :početRamien :dĺžka :úroveň
  ak :úroveň >=1 [
    opakuj :početRamien  [
      do :dĺžka
      vločka :početRamien :dĺžka / 2 :úroveň-1
      vz :dĺžka
      vl 360 / :početRamien
    ]
  ]
koniec
```

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2005_2006/2/riesenia_a_komentare/uloha6.htm

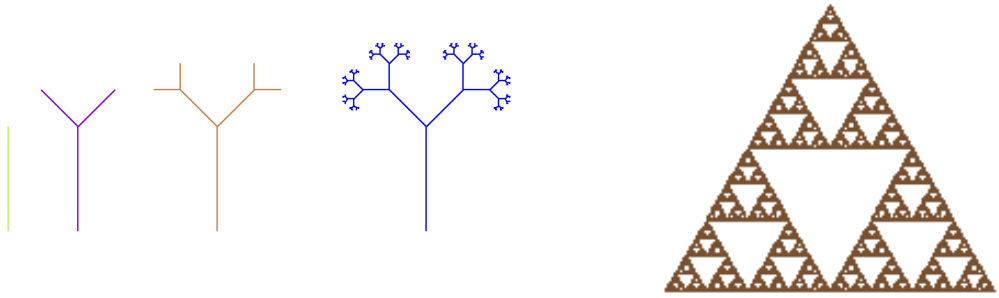
Súťažiaci robili chyby najmä v tom, že vykreslili len prvú úroveň snehovej vločky. Ďalšie, menšie vločky na konci jej ramien nenakreslili vôbec, alebo nenakreslili všetky. Toto bolo spôsobené tým, že rekurzívne volanie sa súťažiaci snažili obísť priamym kreslením menšej vločky.

Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

V súťažnej úlohe sme sa zamerali na použitie jednoduchej rekurzie (1), rozpoznanie samo podobnosti objektov (2), cyklov (3) a podmienok (4) testujúcich úroveň rekurzívneho vnorenia. Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

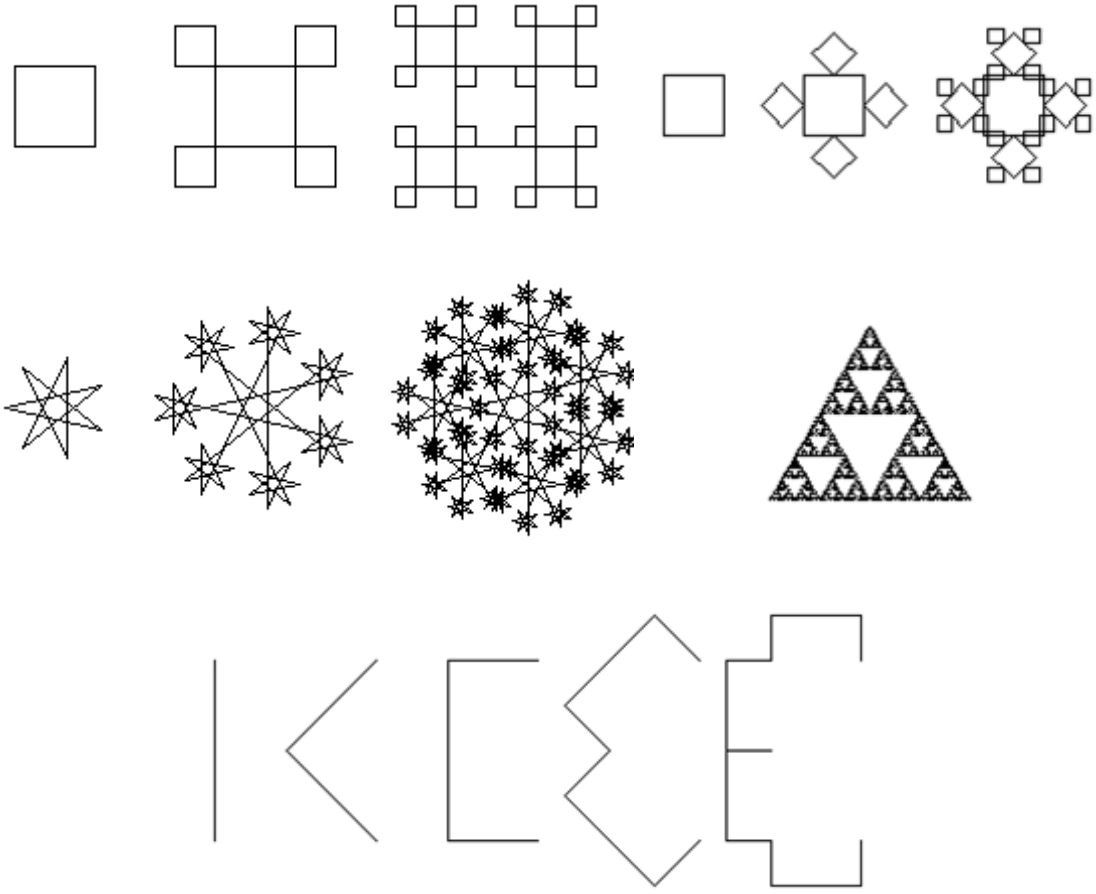
- Analyzujte uvedené obrázky a nájdite časť, ktorá sa v nich opakuje v zmenšenej podobe. Odhadnite, ako vznikli. (1, 2, 3, 4)



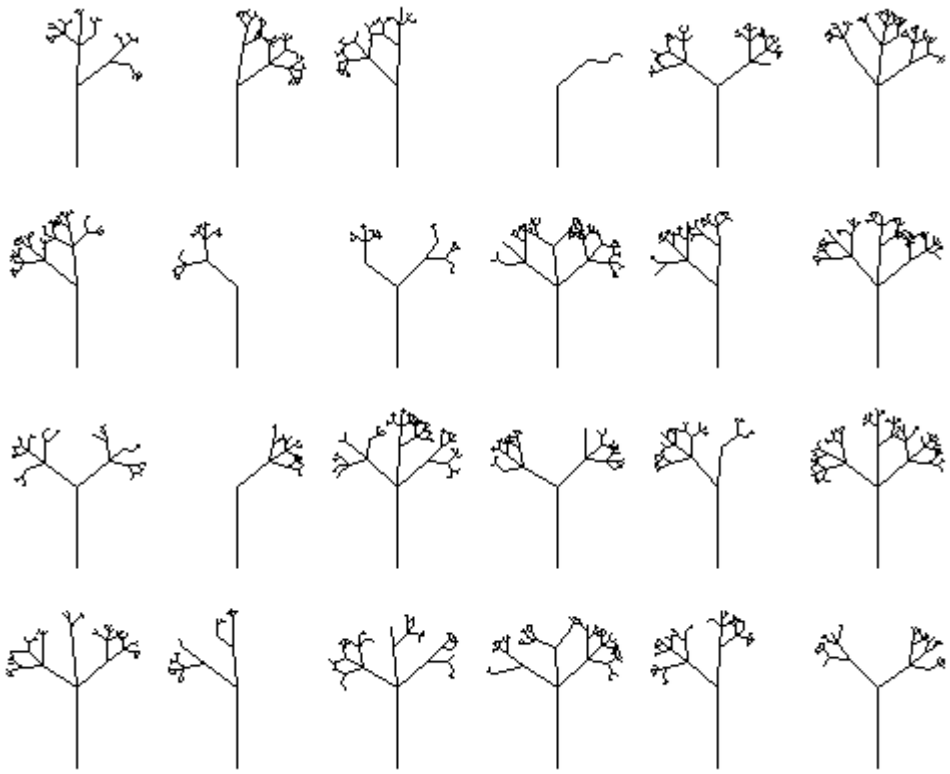
Rozširujúce úlohy:

Rekurzia predstavuje pre túto vekovú kategóriu žiakov pomerne náročnú a abstraktnú techniku. Pri rozširujúcich úlohách sme preto zámerne volili úlohy s grafickým výstupom.

- Vytvorte procedúry na kreslenie rekurzívnych obrázkov. Experimentujte s parametrami a ich hodnotami.



- Rekurzívne obrázky (napr. stromy) sú príliš pravidelné, vyzerajú nereálne. Upravte kreslenie rekurzívneho stromu tak, aby viac zodpovedal realite.



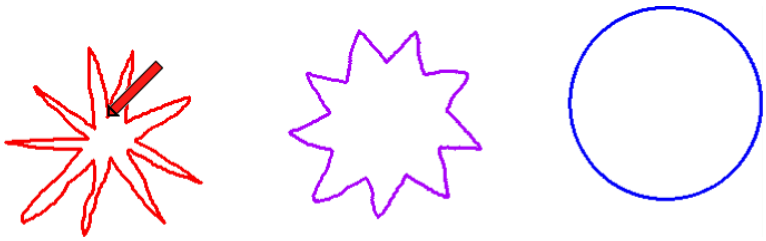
Úloha o prívesku z tvarovaného drôtika

Informácie o úlohe

Úloha PJ 7-4-4 je zameraná na objavenie princípu morfingu s využitím lineárnej interpolácie, precvičenie spracovania dát uložených v dátovej štruktúre zoznam, prácu s vizuálnym komponentom posúvač.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2011_2012/4/4.doc



V Korytnačkove je veľkou módou nosiť prívesky vytvarované z drôtika. Keďže sú korytnačie deti veľmi šikovné, prívesky si vytvárajú samy. Používajú na to veľmi zaujímavý postup. Na papier vľavo nakreslia náčrt tvaru prívesku, napr. hviezdu a vpravo nakreslia nejaký pravidelný útvar, napr. kružnicu. Na obidvoch útvaroch si vyznačia rovnaký počet bodov a určia, ktorý bod z jedného útvaru a ktorý bod z druhého útvaru budú tvoriť pár. Výsledný tvar drôtika medzi oboma útvarmi vznikol tak, že postupne pospájali stredy úsečiek tvorené všetkými určenými pármi bodov z oboch útvarov. Potom vyskúšali spojiť body nie v strede, ale v tretine, štvrtine ... úsečiek. Veľmi sa potešili, keď videli, že keď spoja body v 1/10 úsečky, tak sa výsledný útvar bude podobať viac na prvý útvar, v prípade spojenia bodov v 9/10 úsečky sa bude výsledný útvar podobať viac na druhý útvar. Hovoria, že vzniknutý útvar je dieťaťom obidvoch skôr vykreslených útvarov – rodičov. Hneď sa pustili do tvorenia programu, pomocou ktorého vedia už vykresliť ľubovoľný tvar prívesku a tiež niektorý z pravidelných útvarov (napr. kružnica, úsečka, štvorec). Ostáva im to najdôležitejšie – vymyslieť postup na vykreslenie výsledného útvaru pomocou dvoch už vykreslených útvarov.

Tvojou úlohou je otvoriť Imagine projekt **drotiky.imp** a v ňom naprogramovať procedúru `kresliPotomka`, ktorá vykreslí výsledný útvar (potomka) na základe dvoch už nakreslených útvarov (rodičov) a nastavenia bežca posúvača.

Procedúru ulož ako Imagine projekt s názvom **drotiky.imp**.

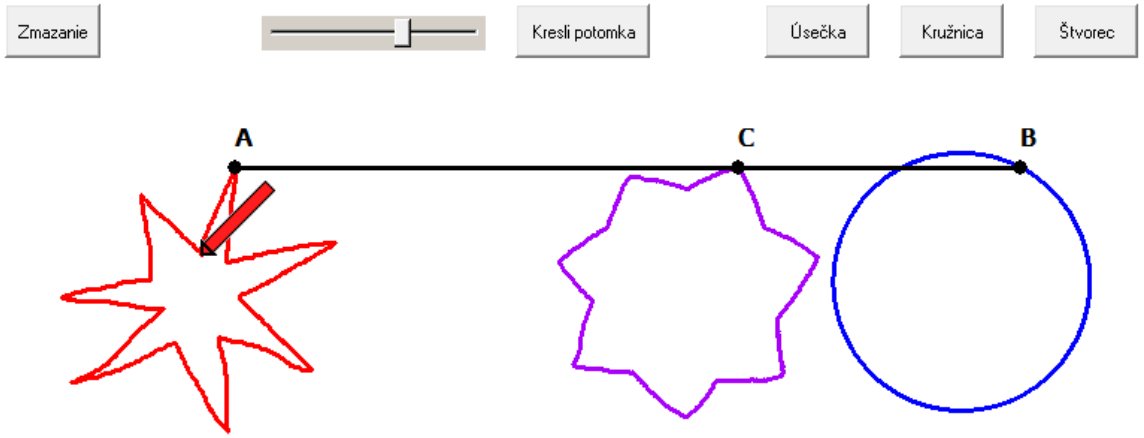
Poznámka: Nakreslené útvary (rodičia) predstavujú lomené čiary, ktoré sú opísané pomocou zoznamov súradníc uložených v globálnych premenných `"ciara1` *a* `"ciara2`*. Posúvačom* `posun`

meníme tvar potomka, pri hodnote 50 je potomok vytvorený uprostred medzi rodičmi, pri hodnote 0 je potomok totožný s prvým rodičom, pri hodnote 100 s druhým rodičom.

Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2011_2012/4/riesenia_komentare/4_riesenie.imp

Podľa zadania úlohy máme vykresliť nový útvar medzi dvoma už nakreslenými útvarmi. Pri polohe posúvača úplne vľavo bude nový útvar totožný s útvarom vľavo, pri pravej krajnej polohe posúvača bude nový útvar totožný s útvarom vpravo. V ostatných polohách posúvača sa bude nový útvar podobať viac na ľavý alebo pravý nakreslený útvar.



Obrázok 15 – Ukážka prechodu medzi tvarmi dvoch útvarov

Nový útvar vytvoríme pomocou bodov oboch pôvodných útvarov. Obidva útvary predstavujú lomené čiary s rovnakým počtom krajných bodov úsečiek. Jednotlivé body nového útvaru budú ležať na úsečke medzi odpovedajúcimi si bodmi oboch pôvodných útvarov. Ich súradnice vypočítame pomocou súradníc bodov obidvoch útvarov a hodnoty posúvača.

Označme si písmenom *C* bod nového útvaru, písmenom *A* bod ľavého útvaru a písmenom *B* bod pravého útvaru. Poďme vypočítať súradnice bodu *C*. Uvažujme len jeho x-ovú súradnicu c_x. Ak poloha posúvača je úplne vľavo (jeho hodnota *posun=0*), hodnota c_x bude rovnaká ako hodnota a_x, t. j. $c_x = a_x$. Ak poloha posúvača je úplne vpravo (jeho hodnota *posun=1*), hodnota c_x bude rovnaká ako hodnota b_x, t. j. $c_x = b_x$. Ak poloha posúvača je niekde medzi ľavým a pravým okrajom (jeho hodnota *posun* je nejaké číslo medzi 0 a 1), hodnotu c_x vypočítame tak, že k hodnote a_x pripočítame určitú časť dĺžky úsečky AB. Túto časť bude určovať hodnota posúvača posun. Hodnotu c_x vypočítame podľa vzťahu $c_x = a_x + (b_x - a_x) \cdot posun$.

Po tomto rozbore môžeme prejsť k vytvoreniu procedúry na vykreslenie nového útvaru. Pomocou cyklu prejdeme oboma zoznamami bodov popisujúcich ľavý a pravý útvar, ktoré sú uložené v globálnych premenných `"čiara1` a `"čiara2`. Keďže obidva zoznamy majú rovnaký počet

prvkov, na ich prechádzanie môžeme použiť cyklus opakovania opakuj, ktorý sa bude opakovať `počet :čiara1` krát alebo `počet :čiara2` krát. Pred samotným cyklom nastavíme korytnačku `"k1` ako aktívnu korytnačku a tiež farbu a hrúbku jej pera. Pre urýchlenie vykresľovania korytnačku skryjeme.

V cykle postupne vyberáme do premenných `"a` a `"b` súradnice odpovedajúcich párov bodov z oboch útvarov. Napr. pre body z prvého útvaru to dosiahneme nasledovne: `urobTu "a prvok počítadlo :čiara1` a pre body z druhého útvaru `urobTu "b prvok počítadlo :čiara2`. Premenné `"a` a `"b` sú dvojprvkové zoznamy, ktoré v prvom prvku zoznamu majú uloženú x-ovú súradnicu a v druhom (poslednom) prvku zoznamu y-ovú súradnicu. Potom x-ovú súradnicu bodu nového útvaru vypočítame pomocou príkazu `urobTu "x (prvý :a) + ((prvý :b) - (prvý :a)) * posun'hodnota / 100`. V danom príkaze sme hodnotu časti výrazu predelili hodnotou 100, lebo hodnota posúvača nebola v rozmedzí hodnôt 0 až 1, ale v rozmedzí 0 až 100. Podobne y-ovú súradnicu bodu nového útvaru určíme pomocou príkazu `urobTu "y (posledný :a) + ((posledný :b) - (posledný :a)) * posun'hodnota / 100`. Napokon bod nového útvaru s takto vypočítanými súradnicami **[x y]** zobrazíme.

Výsledná procedúra `kresliPotomka` môže vyzerať nasledovne:

```
viem kresliPotomka
  odteraz "k1
  skry
  nechFarbaPera "fialová
  nechHrúbkaPera 3
  peroHore

  opakuj počet :čiara1 [
    urobTu "a prvok počítadlo :čiara1
    urobTu "b prvok počítadlo :čiara2
    urobTu "x (prvý :a) + ((prvý :b) - (prvý :a)) * posun'hodnota / 100
    urobTu "y (posledný :a) + ((posledný :b) - (posledný :a)) *
posun'hodnota / 100
    nechPoz zoznam :x :y
    peroDolu
  ]
koniec
```

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2011_2012/4/riesenia_komentare/uloha4.htm

Súťažiaci nás príjemne prekvapili pri riešení tejto úlohy z finálového kola, ktorú až na jeden tím vyriešili ostatní na plný počet bodov. Riešenia boli veľmi rôznorodé, pri prechádzaní jednotlivými bodmi útvarov súťažiaci použili rôzne druhy cyklov (`opakuj`, `kým`, `prePrvky`, `preČísla`). V niektorých riešeniach súťažiaci pri výpočte súradníc bodov nového útvaru nepracovali zvlášť s jednotlivými zložkami súradníc bodu, ale z celým zoznamom určujúcim pozíciu bodu.

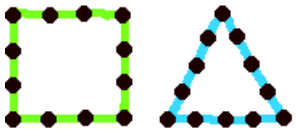
Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

Podstatou riešenia súťažnej úlohy je použitie lineárnej interpolácie – rovnice úsečky (1), spracovanie dát uložených v dátovej štruktúre zoznam (2), práca s vizuálnym komponentom posúvač (3).

Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

- Na obrazovke máme na náhodných pozíciách umiestnené dve korytnačky.
    - Umiestnite tretiu korytnačku na pozíciu, ktorá bude v prostriedku medzi oboma korytnačkami. (1)
    - Umiestnite ďalšie tri korytnačky na pozície, ktoré budú v prvej, druhej a tretej štvrtine veľkosti úsečky určenej pozíciami prvej a druhej korytnačky. (1)
    - Umiestnite tretiu korytnačku na pozíciu, ktorá bude závisieť od polohy bežca posúvača, t. j. v jednej krajnej polohe bežca posúvača sa stotožní s prvou korytnačkou a v úplne pravej polohe s druhou korytnačkou. (1, 3)
- Vykreslite na obrazovke štvorec a trojuholník, ktoré budú reprezentované zoznamami so súradnicami 12 bodov. Vypočítajte súradnice nového útvaru („potomka“ štvorca a obdĺžnika), ktorý bude mať podobu uzavretej lomenej čiary s 12 bodmi. Súradnice týchto bodov budú v strede úsečky medzi odpovedajúcimi dvojicami bodov, jedného bodu zo štvorca a jedného bodu z trojuholníka. (1, 2, 3)



Rozširujúce úlohy:

Súťažnú úlohu môžeme rozšíriť tak, že necháme používateľovi vykresliť vlastnou rukou dva alebo viaceré útvary, ktoré necháme morfovať; prípadne necháme animovať morfovanie jedného textu do druhého textu, resp. jednej fotografie do druhej fotografie.

- Pre 2 (resp. 3) voľnou rukou nakreslené útvary a podľa polohy posúvača (resp. 3 posúvačov) vypočítajte súradnice ďalšieho útvaru („potomka“ nakreslených útvarov) a vykreslite ho na obrazovke.
- Pre dva zadané texty (reprezentované lomenými čiarami) vytvorte animáciu zobrazujúcu postupný prechod jedného textu do druhého textu a naspäť.
- Pre dve fotografie zobrazené na obrazovke vytvorte na voľnom mieste medzi nimi animáciu zobrazujúcu postupný prechod (morfing) jednej fotografie na druhú a naspäť.

Úloha o smere vetra

Informácie o úlohe

Úloha PJ 8-1-3 je zameraná na modelovanie javov (smeru vetra), dôslednú analýzu reálneho problému, použitie celočíselného delenia a zvyšku, prípadne aj použitie dátového typu zoznam.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2012_2013/1/3.doc

V Korytnačkove sú v plnom prúde prípravy na jesennú veternú olympiádu. Korytnáčatá sa rady zapájajú do ich tradičných veterných športov, napr. trojfuku s rozbehom, fuku do výšky a diaľky, chôdzi proti vetru alebo do fúkacieho dvaapolboja.

Pri týchto súťažných disciplínach potrebujú určiť rýchlosť, ale hlavne smer vetra. Všetky korytnáčatá zo školy vedia, že smer vetra sa označuje podľa svetových strán. Rozoznávajú osem smerov vetra – severný, severovýchodný, východný, juhovýchodný, južný, juhozápadný, západný, severozápadný.

Tvojou úlohou je otvoriť Imagine projekt s názvom **vietor.imp** a naprogramovať procedúru `určiSmer`, ktorá pre uhol vetra zadaný v premennej `"uhol` vypíše jeden z ôsmich smerov vetra – S, SV, V, JV, J, JZ, Z, SZ. Procedúru `určiSmer` ulož do Imagine projektu s názvom **vietor.imp**.

Poznámka: Pod uhlom vetra rozumieme uhol, ktorý zviera sever so smerom vetra.

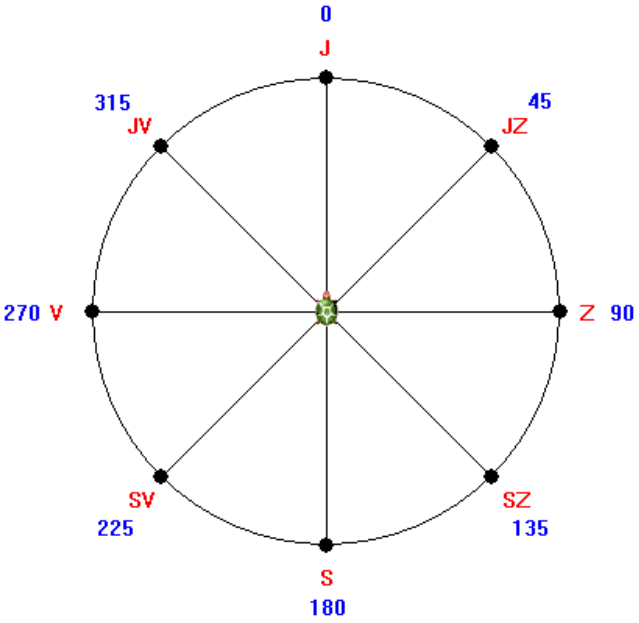
Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2012_2013/1/riesenia_komentare/3_riesenie.imp

Veľmi užitočnou pomôckou pri riešení tejto úlohy je obrázok. Nakreslíme si kruh s rôznymi hodnotami uhlov – násobkami 90 stupňového uhla a potom násobkami 45 stupňového uhla.

Najprv určime, aké uhly odpovedajú svetovým stranám. Sever prislúcha k uhlu 0 stupňov, východ k uhlu 90 stupňov, juh k uhlu 180 stupňov a západ k uhlu 270 stupňov. Ďalším smerom prislúchajú nasledovné uhly – severovýchodu 45, juhovýchodu 135, juhozápadu 225 a severozápadu 315 stupňov.

Pozor na zadanie úlohy, kde sa hovorí, nie KAM fúka vietor, ale ODKIAĽ fúka vietor. Preto, napr. vetru, ktorý fúka zo severu na juh („severáku“) s uhlom 180 stupňov, priradíme označenie severný vietor alebo skratku S.

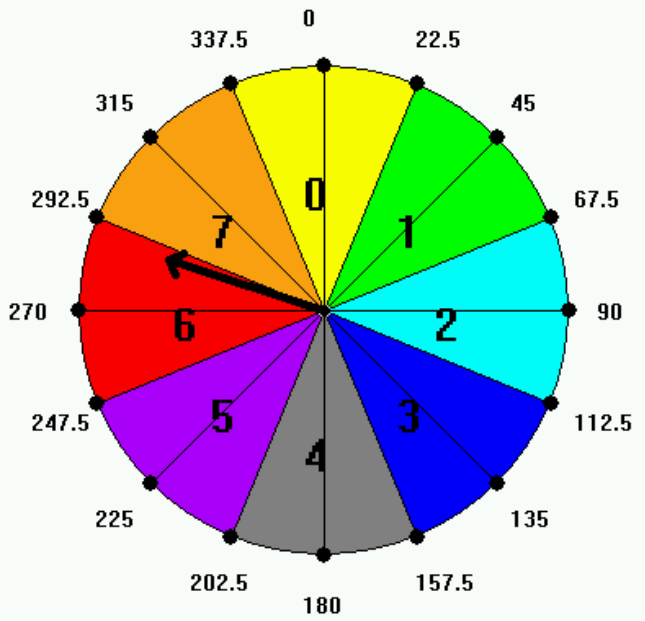


Obrázok 16 – Vzájomné priradenia smeru a názvu vetra

Ak by sme na vstupe mali zadané uhly len ako násobky 45 stupňového uhla, tak riešenie úlohy by bolo veľmi jednoduché. Na základe hodnoty uhla uloženej v premennej "uhol by sme pomocou príkazu vetvenia `akJe` určili patričné označenie vetra. Procedúra `určiSmer` by vyzerala, napr. takto:

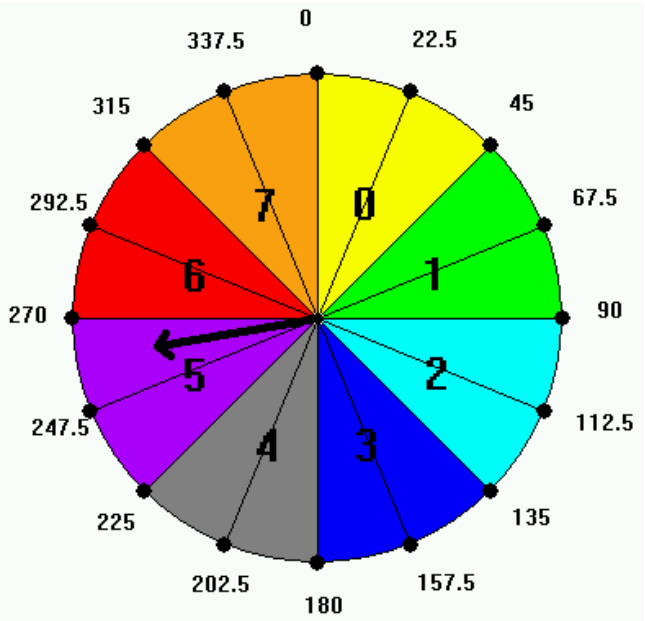
```
viem určiSmer
  akJe :uhol [
      0 [text2'nechHodnota "J ]
     45 [text2'nechHodnota "JZ]
     90 [text2'nechHodnota "Z ]
    135 [text2'nechHodnota "SZ]
    180 [text2'nechHodnota "S ]
    225 [text2'nechHodnota "SV]
    270 [text2'nechHodnota "V ]
    315 [text2'nechHodnota "JV]
  ]
koniec
```

Uhol vetra vo všeobecnosti nemusí byť len celočíselným násobkom 45 stupňového uhla. Ako prevedieme ľubovoľný uhol od 0 do 360 stupňov na jeden z ôsmich požadovaných smerov vetra? Ako určíme hranice medzi, napr. južným a juhozápadným smerom vetra? Prirodzené je určiť hranice v polovici uhlov medzi susednými dvojicami požadovaných smerov. Potom napr. južnému vetru budú odpovedať hodnoty uhla od 337,5 do 22,5 stupňa, juhozápadného vetru hodnoty uhla od 22,5 do 67,5 stupňa, atď. Jednou možnosťou riešenia je zistiť, do ktorého z ôsmich vypísaných rozsahov uhlov patrí zadaný uhol a následne vypísať jeden z ôsmich smerov vetra.



Obrázok 17 – Očíslovanie smerov vetra (1)

Inou možnosťou je očíslovať si požadované smery vetra, napr. číslami od 0 do 7. Pre zadaný uhol vypočítať číslo oblasti, do ktorej patrí daný uhol. A napokon, pre zadané číslo oblasti vypísať odpovedajúci smer vetra. Najprv skúsme uvažovať, ako by sme určili oblasť pre zadaný uhol podľa dolného pomocného obrázka a potom riešenie upravíme, aby vyhovovalo zadaniu úlohy podľa obrázka nad týmto textom.



Obrázok 18 – Očíslovanie smerov vetra (2)

Číslo oblasti na pomocnom obrázku určíme tak, že predelíme zadaný uhol hodnotou 45. Dostaneme desatinné číslo, ktoré je potrebné previesť na celé číslo pomocou funkcie celá. Číslo

oblasti na pomocnom obrázku, do ktorej patrí zadaný uhol, určíme pomocou výrazu: `(celá :uhol /45)`. Číslo oblasti v obrázku nad pomocným obrázkom určíme miernou úpravou výrazu: `(celá (:uhol + 22.5)/45)`. Pre hodnoty uhlov nad 337,5 by sme dostali číslo oblasti 8 namiesto 0. Preto výraz na výpočet oblasti upravíme pomocou funkcie zvyšok nasledovne: `(zvyšok (celá (:uhol+22.5)/45) 8)`.

Na základe tejto úvahy napokon vytvoríme procedúru `určiSmer`:

```
viem určiSmer
  urobTu "oblasť zvyšok (celá (:uhol+22.5)/45) 8
  akJe :oblasť [
    0 [text2'nechHodnota "J]
    1 [text2'nechHodnota "JZ]
    2 [text2'nechHodnota "Z]
    3 [text2'nechHodnota "SZ]
    4 [text2'nechHodnota "S]
    5 [text2'nechHodnota "SV]
    6 [text2'nechHodnota "V]
    7 [text2'nechHodnota "JV]
  ]
koniec
```

Riešitelia, ktorí vedia pracovať so zoznamami, môžu uvedené riešenie skrátiť nasledovne:

```
viem určiSmer1
  urobTu "smery [J JZ Z SZ S SV V JV]
  urobTu "oblasť zvyšok (celá (:uhol+22.5)/45) 8
  text2'nechHodnota prvok :oblasť+1 :smery
koniec
```

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2012_2013/1/riesenia_komentare/uloha3.htm

Úlohu riešilo spolu 28 tímov, bohužiaľ žiaden tím nezískal plný počet bodov. Len 5 tímov pochopilo, že smer vetra sa určuje podľa toho ODKIAĽ fúka vietor, nie KDE fúka vietor. Až 21 tímov použilo príkaz vetvenia neefektívne, kde sa zbytočne veľakrát zisťovala platnosť podmienok. Ukázalo sa, že veľmi málo tímov pozná príkaz vetvenia `akJe` alebo vnorené príkazy `ak2`. Pri určovaní hraníc oblastí sa vyskytli rôzne chyby – posunuté hranice (presne ako sú uvedené na našom pomocnom obrázku), určenie smeru vetra na jednu presnú hodnotu, nie na rozsah čísel, zabudnutie na určenie smeru vetra pre hranice oblastí. Pri tejto a aj pri iných úlohách odporúčame riešiteľom, aby si najprv nakreslili obrázok a na ňom určili hranice oblasti. Zopár tímov použilo pri riešení úlohy zoznamy.

Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

Podstatou riešenia súťažnej úlohy je dôsledná analýza problému nevyhnutná pre správny výpočet (1), použitie celočíselného delenia a zvyšku (2), príkazu vetvenia (3), resp. použitie dátového typu zoznam (4).

Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

- Pre zadaný uhol veterného rukáva, ktorý sú násobkom 90 stupňov, vypočítajte smer vetra. (1, 3)
- Pre zadaný uhol veterného rukáva vypočítajte smer vetra (S, Z, J, V). (1, 2, 3)

Rozširujúce úlohy:

Súťažnú úlohu môžeme rozšíriť tak, že namiesto 8 smerov, budeme uvažovať 16 smerov vetra; prípadne môžeme spracovať merania za určité obdobie (napr. mesiac). Iným rozšírením je úloha s iným kontextom (napr. výpočet bodového skóre zásahov v terči), ktorej výpočet bude vyžadovať nielen uhol, ale aj vzdialenosť od počiatku a tiež použitie dátovej štruktúry zoznam.

- Pre zadaný uhol veterného rukáva vypočítajte smer vetra (S, SSZ, SZ, ZSZ, Z ...).
- Pre zadaný zoznam uhlov, vypočítajte a vytvorte zoznam odpovedajúcich 16 smerov vetra, vykreslite pre každý zo 16 smerov početnosť jeho výskytu v zozname, vytvorte animáciu znázorňujúcu priebeh smerov vetra.
- Vykreslite terč s regiónmi vytvorenými ako prienik zadaného počtu kruhových výsekov a sústredných kružníc. Na základe zadaných bodových hodnôt regiónov vypočítajte pre zadané súradnice zásahov celkové bodové skóre.

Úloha o trojskokanskej súťaži korytnačiek

Informácie o úlohe

Úloha PJ 3-1-3 je zameraná na dôslednú analýzu problému a uvedomenie si, že pri výpočte nemusíme využiť všetky vstupné dáta, na precvičenie vyhodnocovania výrazov a príkazu vetvenia.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2007_2008/1/3.doc

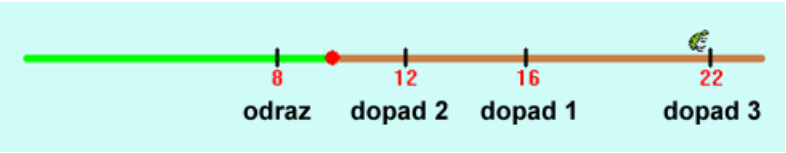
Je jeseň, čas dlho očakávanej korytnačej športovej olympiády. V tomto roku sa v súťaži objavila nová športová disciplína – korytnačí trojskok. Ako vyzerá ihrisko pre korytnačí trojskok? Odrazová doska je vzdialená 10 m od začiatku rozbežiska. Celková dĺžka trojskoku sa meria od odrazovej dosky k miestu dopadu tretieho skoku. Pokus je neplatný, ak skokan urobil prešľap.

Stiahni si projekt **olympiada.imp**, v ktorom je vytvorené prostredie pre túto športovú disciplínu. V premenných `"odraz`, `"dopad1`, `"dopad2`, `"dopad3` sú uložené vzdialenosti odrazu a troch dopadov trojskoku od začiatku rozbežiska. Napíš procedúru `meranie`, ktorá vypíše dĺžku trojskoku alebo informáciu, že pokus bol neplatný. Svoje riešenie ulož ako Imagine projekt pod názvom **olympiada.imp.**

Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2007_2008/1/riesenia_komentare/olympiada_riesenie.imp

Riešenie úlohy nie je ťažké, len si musíme uvedomiť ako vypočítať dĺžku trojskoku. Podľa zadania nám stačí uvažovať len miesto dopadu tretej časti trojskoku (uložené v premennej `"dopad3`) a celkovú dĺžku vypočítať ako rozdiel `:dopad3 - 10`. To samozrejme platí, len v tom prípade, keď miesto odrazu nie je vo väčšej vzdialenosti ako 10 m od začiatku rozbežiska..



Obrázok 19 – Príklad trojskoku korytnačky

Výsledná procedúra vyzerá nasledovne:

```
viem meranie
  ak2 :odraz>10 [
    píš [neplatný pokus]
  ][
    (píš [dĺžka skoku je:] :dopad3 - 10 "m)
  ]
koniec
```

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2007_2008/1/riesenia_komentare/uloha3.htm

Správne vyriešili túto úlohu 4 žiaci (2 profíci a 2 experti), ďalší 15 riešitelia sa dopustili niektorej z uvedených chýb:

- výpočet dĺžky trojskoku bol meraný od miesta odrazu, nie od odrazovej dosky,
- do výpočtu dĺžky trojskoku boli zbytočne zarátané aj vzdialenosti dopadu prvej a druhej časti trojskoku, resp. bol výpočet celkovej dĺžky nesprávny,
- pri ošetrení prešľapu trojskoku, niektorí riešitelia použili overenie nerovnosti `:odraz>9` namiesto `:odraz>10`,
- väčšina riešiteľov namiesto úplného príkazu vetvenia `ak2`, použili dva príkazy vetvenia `ak`.

Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

Podstatou riešenia súťažnej úlohy je dôsledná analýza problému pri výpočte dĺžky skoku (1), uvedomenie si, že pri výpočte nemusíme využiť všetky vstupné dáta (2), použitie príkazu vetvenia (3). Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

- Vypočítajte dĺžku trojskoku z miesta, ak sú zadané dĺžky všetkých troch častí trojskoku – `:dĺžka1`, `:dĺžka2`, `:dĺžka3`. (1)
- Vypočítajte dĺžku trojskoku z miesta, ak sú zadané vzdialenosti všetkých troch častí trojskoku od počiatku – `:dopad1`, `:dopad2`, `:dopad3`. (1, 2)
- Vypočítajte dĺžku skoku do diaľky s rozbehom s odrazovou doskou 10 m od začiatku rozbežiska, ak je zadaná vzdialenosť odrazu (`:odraz`) a vzdialenosť dopadu (`:dopad`) namerané od začiatku rozbežiska. (1, 3)

Rozširujúce úlohy:

Súťažnú úlohu môžeme rozšíriť o spracovanie viacerých trojskokov jedného, či viacerých trojskokanov.

- Vypočítajte maximálnu dĺžku trojskoku trojskokana, ktorý absolvoval 6 pokusov, pričom máme zadaných pri každom pokuse hodnotu odrazu a dopadu trojskoku od začiatku rozbežiska so zadanou vzdialenosťou odrazovej dosky.
- Vyhodnoťte výsledky trojskoku skupiny trojskokanov absolvujúcich kvalifikačné a finálové kolo súťaže.

Úloha o tom, ako korytnačky v bludisku blúdili

Informácie o úlohe

Úloha PJ 5-2-6 je zameraná na nájdenie stratégie prechodu bludiskom a jej naprogramovanie tak, aby bola realizovateľná vykonávateľom – korytnačkou. Žiaci teda riešia dva rozdielne problémy. Prvý problém, nájdenie stratégie, priamo nesúvisí s programovaním a je riešiteľný aj bez počítača. Druhý problém spočíva v implementácii stratégie do programovacieho prostredia. Tu by sme ešte radi uviedli fakt, že v bludisku nejde vždy len o to, ako efektívne sa z neho dostať. Často ide len o zábavu, napr. aj zo stratenia sa v ňom.

Pri samotnej implementácii stratégie sa využíva výsledok analýzy obrázka (resp. jeho časti v blízkom okolí korytnačky). Predpokladá sa použitie podmienok a cyklov. Súčasťou zadania je prostredie, v ktorom sa vygeneruje jedno, z vopred pripravených bludísk.

V autorskom komentári používame analógiu s tmavou chodbou. V tomto prípade nemáme žiadnu globálnu informáciu o bludisku a tak sa musíme spoliehať len na lokálnu informáciu z bezprostredného okolia. Podobne je na tom aj korytnačka v pripravenom prostredí.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2009_2010/2/6.doc

Do Korytnačkova zavítal pojazdný zábavný park. Pre korytnačie deti to je veľká udalosť. A tento rok ešte o niečo väčšia. V zábavnom parku pribudla nová atrakcia, bludisko. Lenže nájsť cestu z bludiska von nie je také jednoduché, ako by sa na prvý pohľad zdalo. Neraz museli zasahovať korytnačí požiarnici a stratené deti z bludiska vytiahnuť žeriavom. Náčelník požiarnikov preto rozhodol, že deti do bludiska môžu vojsť len vtedy, ak poznajú postup, ako sa z neho dostať von.

Vedel by si im pomôcť? Nauč korytnačie deti, ako prejsť bludiskom.

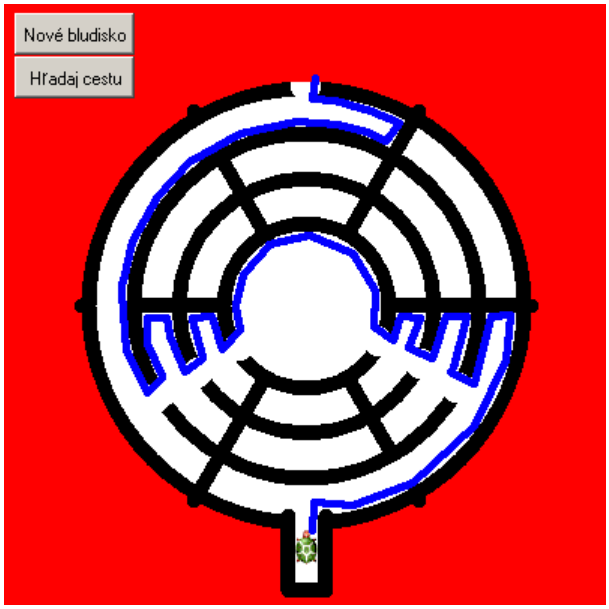
Stiahni si projekt **bludisko.imp**. Tvojou úlohou je naprogramovať procedúru `hľadajCestu`, ktorá vyvedie korytnačku von z bludiska.

Svoje riešenie ulož ako Imagine projekt pod názvom **bludisko.imp**.

Autorské riešenie úlohy

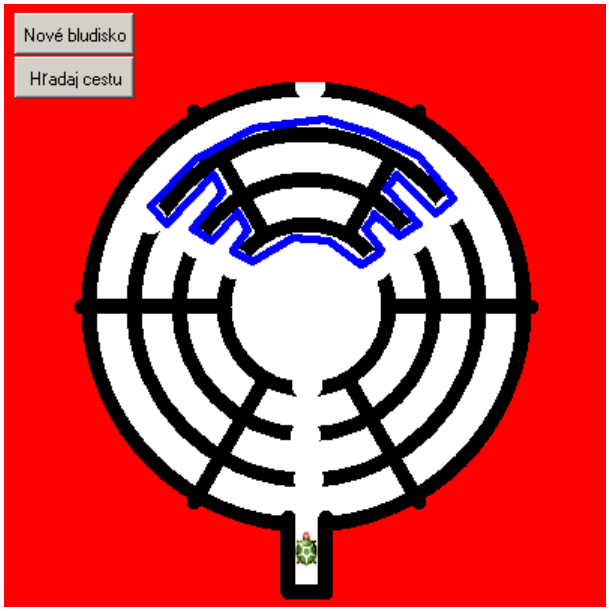
URL = http://di.ics.upjs.sk/palmaj/zadania/2009_2010/2/riesenia_komentare/6_riesenie.imp

Naprogramovanie tejto úlohy je jednoduché. Ale len v tom prípade, ak poznáme postup – stratégiu, ako bludiskom prejsť. Existuje niekoľko postupov ako prejsť bludiskom. Pri ich objavovaní si môžeme pomôcť predstavou, ako by sme postupovali, ak by sme sa ocitli v budove, v ktorej zhasli svetlá. Zrejme by sme začali „šmátrať“ rukami okolo seba a hľadali by sme nejaký oporný bod. Najlepšie stenu. Potom by sme pridŕžajúc sa tejto steny postupovali popri nej a dúfali, že sa takto dostaneme do miesta, kde bude už vidno alebo ešte lepšie, priamo k východu. Zrejme by sme vošli aj do nejakých slepých uličiek a na ich konci by sme sa museli otočiť a vrátiť sa. Stále by sme sa však pridŕžali steny ohraničujúcej jednotlivé chodby. Takýto postup môžeme jednoducho nazvať, napr. pravidlo pravej ruky. Pravou rukou sa dotkneme steny a postupujeme popri nej. Ako by mohlo vyzerať naše blúdenie ukazuje nasledujúci obrázok.



Obrázok 20 – Prechod bludiskom podľa stratégie pravidla pravej ruky

Šikovnejší z vás určite prišli na to, že rovnako dobre by sme mohli použiť aj pravidlo ľavej ruky. A tí ešte šikovnejší prišli aj na to, že nie každá stena je dobrá na to, aby nás vyviedla z bludiska.

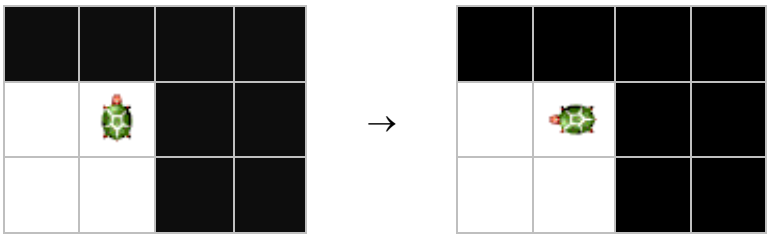


Obrázok 21 – Nevhodný výber steny pre prechode bludiskom

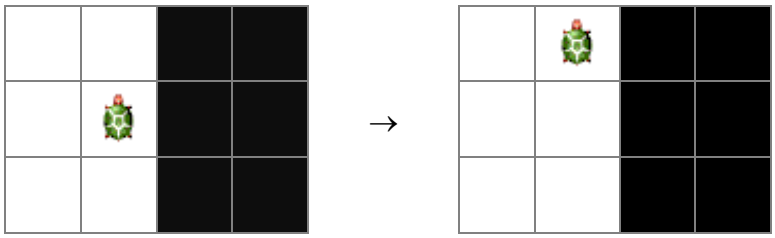
Keďže vchod aj východ z bludiska sú na jeho vonkajšej časti, mali by sme sa pridŕžať vonkajšej steny. Stačí teda, ak korytnačka cúvne k najbližšej stene (keďže je vo vstupe, stena je určite vonkajšia) a otočí sa tak, aby mala stenu po pravej ruke. Potom bude postupovať popri nej až kým nenarazí na červenú farbu pri východe z bludiska. Týmto sme vyriešili, ako sa po bludisku pohybovať smerom k východu. Našli sme stratégiu. Ďalším krokom je túto stratégiu implementovať do prostredia. Korytnačka samozrejme nemá ruky a nevie sa dotknúť steny. Nakoniec, stena ako taká, je zobrazená len ako čierna čiara. Čiara, ktorá je zložená z čiernych bodiek – bodov obrazovky. Takže pridŕžať sa steny v podaní korytnačky znamená, pohybovať sa tak, aby pixel, ktorý je vpravo od nej bol vždy čiernej farby. Začnime teda tým, že sa korytnačka presunie k stene, ktorá je hneď za ňou a otočí sa tak, aby stena (čierny bod) bol od nej vpravo.

```
viem hľadajCestu
  kým [farbabodu <> "čierna] [vz 1]
  do 1 vp 90
koniec
```

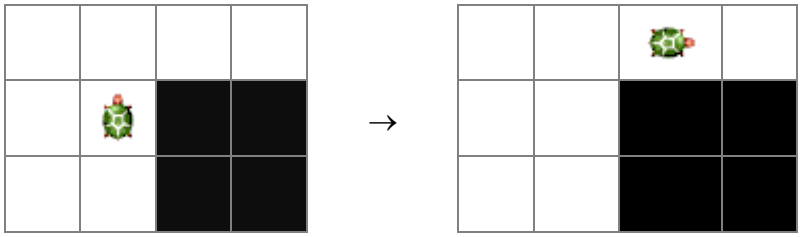
Počas cesty k východu (ak korytnačka používa stratégiu pravidla pravej ruky) sa korytnačka môže nachádzať v niekoľkých rôznych pozíciách vzhľadom k stene. Tieto sú uvedené na nasledujúcich obrázkoch aj s pozíciou korytnačky, ktorú dosiahne, ak sa pokúsi spraviť krok vpred:



Obrázok 22 – Zmena pozície korytnačky v kúte

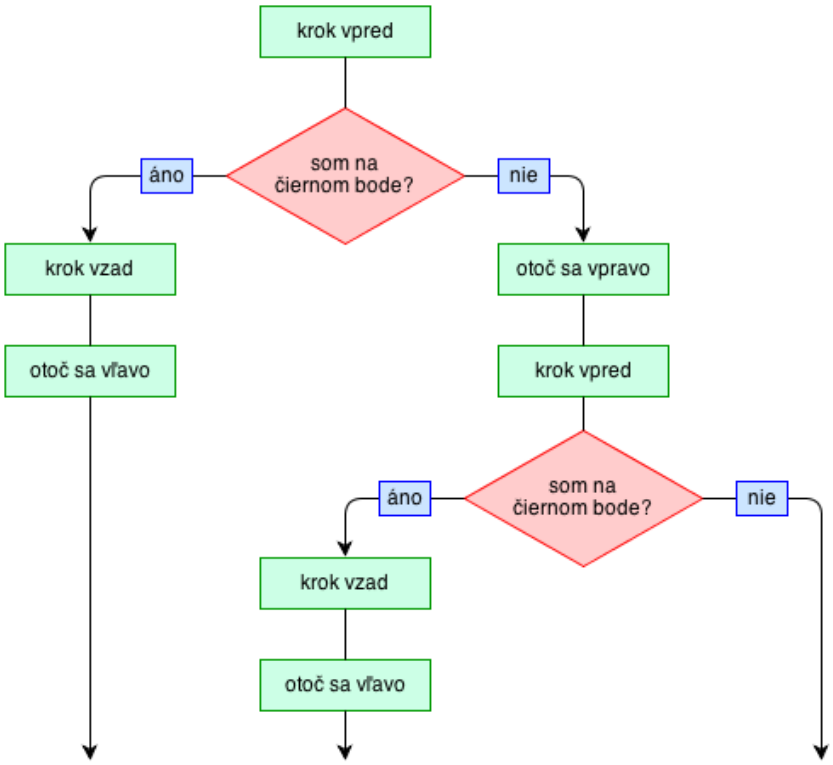


Obrázok 23 – Zmena pozície korytnačky, ak sa pohybuje pozdĺž steny



Obrázok 24 – Zmena pozície korytnačky, ak sa pohybuje okolo rohu steny

Jednotlivé situácie rozpoznáme podľa nasledujúceho postupu:



Obrázok 25 – Rozhodovací strom pri pohybe popri stene

Tento postup bude korytnačka opakovať dovtedy, kým nenájde východ z bludiska. To, že našla východ spoznáme tak, že korytnačka sa dostane na červenú farbu. Výsledná procedúra môže vyzerať nasledovne:

```
viem hľadajCestu
  kým [farbabodu <> "čierna] [
    vz 1
  ]
  do 1 vp 90
```

```
  kým [farbabodu <> "červená] [
    do 1
    ak2 farbabodu = "čierna [
      vz 1 vl 90
    ][
      vp 90 do 1
      ak farbabodu = "čierna [vz 1 vl 90]
    ]
    čakaj 1
  ]
koniec
```

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2009_2010/2/riesenia_komentare/uloha6.htm

Žiaci sa často snažili kontrolovať obe strany bludiska súčasne, čo spôsobilo, že riešenie vyzerá chaoticky, resp. je neefektívne, pretože sa korytnačka často vracia na rovnaké miesta. Niektorí zakomponovali do svojho riešenia náhodnosť, čo v tomto prípade nie je najlepšia voľba. Korytnačka totiž môže v bludisku blúdiť veľmi dlho.

Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

Súťažná úloha je zameraná na nájdenie stratégie (1) prechodu bludisko. Počas pohybu korytnačky v bludisku sa predpokladá, že bude analyzovať svoje okolie – obrázok (2) a podľa výsledku lokálnej analýzy postupovať ďalej. Sekundárne je úloha zameraná na korytnačiu geometriu (3), cykly (4) a testovanie podmienok (5). Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

- Naprogramujte korytnačku tak, aby vypísala farbu bodu, na ktorom sa nachádza. (2)
- Naprogramujte korytnačku tak, aby sa pohybovala vpred až kým nenarazí na bod čiernej farby. (2, 3, 4, 5)
- Naprogramujte korytnačku tak, aby sa pohybovala vpred, až kým nenarazí na bod inej farby než ten, na ktorom začal jej pohyb. (2, 3, 4, 5)
- Naprogramujte korytnačku, ktorá sa nachádza v uzatvorenej oblasti tak, aby sa pohybovala podľa nasledovných pravidiel:
    - korytnačka sa otočí sa o náhodný uhol a spraví krok vpred,
    - ak sa dostala mimo uzatvorenú oblasť, spraví krok vzad. (1, 2, 3, 4, 5)

Rozširujúce úlohy:

Súťažnú úlohu môžeme modifikovať tak, že budeme požadovať náročnejšiu analýzu (napr. nie len blízke okolie) alebo zmenou parametrov bludiska.

- Naprogramujte korytnačku tak, aby zistila, či sa nachádza v uzatvorenej oblasti (oblasť ohraničená uzatvorenou krivkou).
- Naprogramujte korytnačku tak, aby počas prechodu bludiskom (ako v úlohe o blúdení korytnačiek v bludisku) vykresľovala trajektóriu svojho pohybu.
- Naprogramujte korytnačku tak, aby našla východ z bludiska aj v prípade, že vstup alebo výstup sa nenachádzajú na okraji bludiska.

Úloha o tom, ako Korypolícia kontroluje dodržiavanie dopravných predpisov

Informácie o úlohe

Úloha PJ 4-3-4 je zameraná na spracovanie dát a ich vyhodnocovanie. Pri hľadaní riešenia využívame stratégiu riešenie problémov – vytvor si tabuľku. Dáta sú uložené v premennej typu zoznam a majú vopred danú štruktúru a význam. Úlohou žiakov je tieto dáta prečítať, analyzovať a vyhodnotiť.

Zadanie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2008_2009/3/4.doc

V súčasnosti takmer každá korytnačia rodinka vlastní minimálne jedno auto. Korypolícia tak začala riešiť narastajúci počet dopravných priestupkov a nehôd. Dopravné predpisy v Korytnačkove pritom nie sú nijako komplikované:

- v obci je stanovená maximálna rýchlosť na 40 km/h,
- na diaľnici je stanovená maximálna rýchlosť na 90 km/h,
- na ostatných cestách (teda mimo obce a mimo diaľnic) je stanovená maximálna rýchlosť na 65 km/h,
- žiadnou obcou nevedie diaľnica.

Korypolícia sa preto rozhodla namontovať do každého auta v Korytnačkove špeciálny prístroj, ktorý zaznamenáva priebeh jazdy a vďaka tomu môže Korypolícia kedykoľvek skontrolovať, ako jednotliví vodiči jazdili. Prístroj zaznamenáva dopravné značky označujúce začiatok (`zo`) a koniec (`ko`) obce, dopravné značky označujúce začiatok (`zd`) a koniec (`kd`) diaľnice a pri každej zmene rýchlosti zaznamenáva aktuálnu rýchlosť vozidla. Jazda každého auta začína v obci.

Stiahni si Imagine projekt **rychlost_jazdy.imp**. V projekte sa nachádza jedno tlačidlo v podobe oranžového auta. Kliknutím na toto tlačidlo sa vygeneruje zoznam **z** obsahujúci záznam jazdy a spustí sa procedúra `kontrola`, ktorá zistí a vypíše, či vodič auta porušil alebo neporušil dopravné predpisy. Tvojou úlohou je vytvoriť túto procedúru `kontrola`.

Svoje riešenie ulož ako Imagine projekt pod názvom **rychlost_jazdy.imp**.

Autorské riešenie úlohy

URL = http://di.ics.upjs.sk/palmaj/zadania/2008_2009/3/riesenia_komentare/riesenie_rychlost_jazdy.imp

Vymyslieť riešenie tejto úlohy nie je ťažké. Budeme však musieť trochu porozmýšľať nad tým, ako otestovať porušenie (alebo neporušenie) dopravných predpisov a ako tieto testy šikovne zapísať.

Najskôr sa pozrime na všetky možné situácie, ktoré môžu nastať. Vodič sa môže nachádzať na troch typoch ciest (obec, mimo obce, diaľnica) a jeho rýchlosť môže byť zaradená do štyroch intervalov (povolené v obci, povolené mimo obce, povolené na diaľnici, úplne nepovolená). Všetky možné situácie sú znázornené v tabuľke:

miesto \ rýchlosť	<0 ,40>	(40,65>	(65, 90>	(90, ∞)
Obec	✓	✗	✗	✗
Mimo obce	✓	✓	✗	✗
Diaľnica	✓	✓	✓	✗

Červené krížiky znázorňujú situácie, v ktorých vodič porušil predpisy. Stačí teda otestovať, či nastala niektorá z nich.

Záznam o priebehu jazdy obsahuje informácie o tom, kde sa vodič nachádzal (v tabuľke sú to riadky) a akú mal rýchlosť (v tabuľke sú to stĺpce).

Postupne budeme prechádzať záznam o priebehu jazdy. Ak nájdeme záznam o zmene miesta vodiča (`zo`, `ko`, `zd`, `kd`), zapamätáme si nové miesto. Ak nájdeme záznam o rýchlosti (číselná hodnota) otestujeme porušenie predpisov. Nezabudnime na to, že každá jazda začína v obci. Výsledná procedúra by mohla vyzerať nasledovne:

```
viem kontrola
  urobTu "miesto "obec

  prePrvky "záznam :z [
    akJe :záznam [
      zo    [urobTu "miesto "obec]
      ko kd [urobTu "miesto "mimoobec]
      zd    [urobTu "miesto "diaľnica]
            [
              ak zároveň :miesto = "obec :záznam > 40 [
                píš "prekročil ukonči
              ]
              ak zároveň :miesto = "mimoobec :záznam > 65 [
                píš "prekročil ukonči
              ]
              ak :záznam > 90 [
                píš "prekročil ukonči
              ]
            ]
    ]
  ]
  píš "neprekročil
koniec
```

Všimnime si ešte jednu vec. Ak zistíme, že vodič prekročil rýchlosť, nemusíme záznam jazdy ďalej prezerať. Procedúra vypíše informáciu o prekročení rýchlosti a ukončí sa príkazom `ukonči`. To je aj dôvod na to, že sme nepoužili vnorené podmienky.

Komentár k žiackym riešeniam

URL = http://di.ics.upjs.sk/palmaj/zadania/2008_2009/3/riesenia_komentare/uloha4.htm

Pri riešení tejto úlohy robili žiaci nasledovné chyby:

- Žiaci vyhodnocovali dáta zo zoznamu aj potom, čo zistili prekročenie rýchlosti. Nie je to síce priamo chyba, ale je to zbytočné a takéto riešenie nie je efektívne.
- Žiaci vypisovali správu po prečítaní každého prvku zoznamu. Ak vodič prekročil rýchlosť, stačilo to vypísať len raz a s vyhodnocovaním skončiť. Ak sa vyhodnotil celý zoznam a nezistili sme prekročenie rýchlosti, až potom stačí vypísať túto informáciu.
- Niektorí žiaci použili rekurziu, ktorá je v tomto prípade zbytočná (najmä pre svoju náročnosť na pamäť).

Námety na prípravné a rozširujúce úlohy

Prípravné úlohy:

Podstatou riešenia úlohy je analýza dát (1) uložených v dátovej štruktúre zoznam (2). Úlohou žiakov bolo navrhnúť pravidlá (3) podľa ktorých vyhodnocovať dáta. Pri riešení tejto úlohy sme použili stratégiu riešenie: vytvor si tabuľku (4). Na precvičenie týchto elementov učiva navrhujeme zaradiť nasledovné prípravné úlohy:

- Vytvorte procedúru, ktorá postupne vypíše prvky zo zadaného zoznamu. (2)
- V zozname sú uložené nadmorské výšky bodov, ktoré prístroj zaznamenal v určených časových intervaloch počas turistického výletu.
    - Vytvorte procedúru, ktorá vypíše výšku najvyššieho dosiahnutého miesta. (1, 2, 3)
    - Vytvorte procedúru, ktorá vypíše nadmorské výšky všetkých dosiahnutých vrcholov (za vrchol považujeme také miesto, na ktoré sme vystúpili a z ktorého sme zostúpili, inak povedané, lokálne maximum). (1, 2, 3)
    - Vytvorte procedúru, ktorá vypíše hodnotu celkového stúpania a celkového klesania. (1, 2, 3)
- Karol začal trénovať na vytrvalostné preteky. Svoj tréningový plán rozdelil na týždňové úseky a dĺžku behu v jednotlivé dni si naplánoval takto:
    - prvý deň bude bežať 12 minút,
    - počas ďalších 6 dní týždňa bude tento čas zvyšovať o minútu denne,
    - každý nový týždeň začne s časom, ktorý je o 3 minúty kratší ako čas posledného dňa predchádzajúceho týždňa a v nasledujúce dni týždňa bude čas behu po minúte zvyšovať.

Vytvorte procedúru ktorá zistí, ako dlho má Karol bežať v zadanom dni jeho prípravy na vytrvalostné preteky. (4)

Rozširujúce úlohy:

Náročnejšie úlohy podobného typu môžeme získať tak, že žiakom poskytneme na analýzu štruktúrovanejšie dáta.

- Naprogramujte korytnačku tak, aby vykreslila obrázky, ktorých popis je uložený v zozname. Napr. `[[štvorec 100] [trojuholník 60] ...]` (útvar so zadanou dĺžkou strany) Je možné doplniť aj ďalšie útvary.
- Vytvorte procedúru, ktorá vykreslí obrázok ktorého popis (návod na vykreslenie) je zadaný v zozname. Zoznam má nasledovnú štruktúru:
  - návod → []
  - návod → [útvar]
  - útvar → útvar útvar
  - útvar → [koľkoKrát koľkoDopredu uholOtočenia]
- V zozname je uložený stav rozohranej hry piškvorky (3×3). Vytvorte procedúru ktorá zistí, či niektorý z hráčov vyhral a ak áno, ktorý.

  Napr.:

`[[1 0 2][0 1 2][0 2 1]]` → | o | | x |
| | o | x |
| | x | o | → Vyhral hráč 1

Námety úloh a prieskumných otázok pre učiteľov a autorov úloh:

- Pre niektorú z uvedených súťažných úloh vytvorte alternatívne prípravné a rozširujúce úlohy.
- Dajte svojim žiakom vyriešiť prípravné aj rozširujúce úlohy niektorej z uvedených súťažných úloh. Ako sa im darilo? Našli sa aj takí žiaci, ktorí vyriešili aj niektorú z rozširujúcich úloh?

ZÁVER

Problematika tvorby úloh je pre učiteľa informatiky kľúčovou pri rozvíjaní jeho plánovacích učiteľských kompetencií. Napriek tomu chýba odborná literatúra z oblasti didaktiky informatiky zameraná na tvorbu úloh, resp. tvorbu prípravných a rozširujúcich úloh k vybraným úlohám.

Vychádzajúc z teoretických východísk tvorby úloh (komponenty úlohy, zameranie úlohy, typy formulácií úloh, tvorba systémov úloh) sme na reprezentatívnej vzorke úloh súťaže PALMA junior mapujúcej jednotlivé oblasti (programovanie, algoritmy, matematika) ukázali viaceré stratégie riešenia algoritmických problémov. Pri každej úlohe uvádzame zoznam jej učebných cieľov a zistené miskoncepcie žiakov získané analýzou žiackych riešení. Na základe cieľov a zistených miskoncepcií uvádzame zoznam prípravných úloh. Danú súťažnú úlohu nepovažujeme za konečný cieľ, ale uvádzame námety ďalších úloh rozširujúcich učebné ciele pôvodnej úlohy. Pôvodná úloha spolu s prípravnými a rozširujúcimi úlohami tvoria „rodinu“ úloh, v ktorej sú vzťahy medzi jej členmi určené primárne učebnými cieľmi. Vďaka možnosti meniť kontext úloh v rámci „rodiny“ rozvíjame žiakov aj v iných oblastiach vzdelávania, napr. geografia, fyzika, umenie. Celkovo rozvíjame u žiakov nielen ich matematickú gramotnosť, ale aj prírodovednú a čitateľskú gramotnosť.

Veríme, že túto monografiu využijú didaktici informatiky, autori úloh programátorských súťaží a učitelia informatiky pri práci s talentovanou mládežou.

Ďakujeme vedeckému redaktorovi doc. RNDr. Stanislavovi Lukáčovi, PhD. a recenzentom doc. RNDr. Gabriele Lovászovej, PhD. a RNDr. Róbertovi Hajdukovi, PhD. za dôsledné prečítania rukopisu monografie a za ich komentáre a pripomienky, ktorými prispeli veľkou mierou ku kvalite tejto monografie. Rovnako ďakujeme Agentúre pre podporu výskumu a vývoja za podporu vydania tejto monografie v rámci projektov APVV-0057-09 Rozvíjanie talentu prostredníctvom korešpondenčných seminárov a súťaží a APVV-0715-12 Výskum efektívnosti metód inovácie výučby matematiky, fyziky a informatiky.

LITERATÚRA

- (BELUŠOVÁ et al., 2002) BELUŠOVÁ, Mária – VARGA, Mário – ZIMANOVÁ, Ružena: Informatika pre stredné školy: Algoritmy s Pascalom. Bratislava: SPN, 2002. ISBN 80-08-03289-8.
- (BLAHO – KALAŠ, 2006) BLAHO, Andrej – KALAŠ, Ivan: Tvorivá informatika: 1. zošit z programovania. Bratislava: SPN – Mladé letá, 2006. ISBN 80-10-00019-1.
- (GUNIŠ et al, 2007) GUNIŠ, Ján – ŠNAJDER, Ľubomír – GUNIŠOVÁ, Valentína – SZABOÓVÁ, Zuzana – HRICOVÁ, Andrea: PALMA junior – programming competition in Imagine. In: Proceeding of the 11th European Logo Conference EuroLogo 2007 Bratislava. Knižničné a edičné centrum FMFI UK, Bratislava 2007. pp. 54. ISBN 978-80-89186-20-4.
- (GUNIŠ – ŠNAJDER, 2009) GUNIŠ, Ján – ŠNAJDER, Ľubomír: Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika – Tvorba úloh a hodnotenie žiakov v predmete informatika. 1. vydanie. Bratislava: Štátny pedagogický ústav v Bratislave, 2009. 40 s. ISBN 978-80-8118-012-5.
- (GUNIŠ – ŠNAJDER, 2013) GUNIŠ, Ján – ŠNAJDER, Ľubomír: Súťaž PALMA junior – pravidlá súťaže, zadania, autorské riešenia, komentáre súťažných riešení, aktuálne poradie, učebné texty, prihláška. Dostupné na internete: http://di.ics.upjs.sk/palmaj/
- (KALHOUS – OBST, 2002) KALHOUS, Zdeněk – OBST, Otto a kol.: Školní didaktika. 1. vyd. Praha: Portál, 2002. 448 s. ISBN 80-7178-253-X.
- (KOPKA, 2006) KOPKA, Jan: Zkoumání ve školské matematice. Ružomberok: Katolícka univerzita v Ružomberku, 2006. 151 s. ISBN 80-8084-064-4.
- (LIPKOVÁ, 2009) LIPKOVÁ, Juliana: Informatické súťaže na Slovensku. [elektronický dokument]. In: Zborník konferencie Didinfo 2009 (CD ROM), s. 129 – 131. Banská Bystrica: Univerzita Mateja Bela, ISBN 278–80–8083–720–4.
- (LUKÁČ, 1999) LUKÁČ, Stanislav: Počítačom podporovaná zbierka úloh z programovania. Matematika – fyzika – informatika 9 (1999/2000), č. 3, s. 172 – 182, ISSN 1210-1761.
- (MIKOLAJOVÁ, 2012) MIKOLAJOVÁ, Katarína: Začíname programovať v prostredí Scratch – učebný materiál pre žiakov 5. – 7. ročníka ZŠ. [elektronický dokument]. In: Zborník konferencie Didinfo 2012 (CD ROM), s. 147 – 156. Banská Bystrica: Univerzita Mateja Bela, ISBN 978-80-557-0342-8.
- (PROJECT MATH, 2010) Project Maths – Learning and Teaching for the 21st Century. Problem Solving Strategies. Dostupné na internete: http://www.projectmaths.ie/workshops/workshop_3_NR/Problem%20Solving%20Posters%20english.pdf

- (ŠIŠKOVÁ, 2010) ŠIŠKOVÁ, Juliana: Prehľad informatických súťaží. Dostupné na internete: http://people.ksp.sk/~julka/sutaze/
- (ŠNAJDER – GUNIŠ – GUNIŠOVÁ, 2008) ŠNAJDER, Ľubomír – GUNIŠ, Ján – GUNIŠOVÁ, Valentína: Methodology design of algorithm development teaching based on content analysis of pupils' solutions. In: The 3rd International Conference – ISSEP 2008 Informatics in Secondary Schools – Evolution and Perspective July 1 – 4, 2008, Torun, Poland. Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, Poland 2008. p. 20 – 29, ISBN 978-83-60425-31-2.
- (ŠVEC – FILOVÁ – ŠIMONÍK, 2004) ŠVEC, Vlastimil – FILOVÁ, Hana – ŠIMONÍK, Oldřich: Praktikum didaktických dovedností. Brno: Masarykova univerzita v Brně, Pedagogická fakulta, 2004. 90 s., ISBN 80-210-2698-7.
- (ŠVEDA, 1992) ŠVEDA, Dušan: Tvorba systémov úloh v matematike. Prešov: MC v Prešove, 1992. 43 s., ISBN 80-85410-36-2.
- (VARGA – BLAHO – ZIMANOVÁ, 1999) VARGA, Mário – BLAHO, Andrej – ZIMANOVÁ, Ružena: Informatika pre gymnáziá: Algoritmy s Logom. Bratislava: SPN – Mladé letá, s. r. o., 1999. ISBN 80-08-02965-X.

Tvorba úloh pre programátorské súťaže

Autori:	RNDr. Ľubomír Šnajder, PhD. , PaedDr. Ján Guniš, PhD.
Rozsah strán:	79
Rozsah AH:	4,4
Vydavateľ:	Univerzita Pavla Jozefa Šafárika v Košiciach
Vydanie:	prvé
Rok vydania:	2014
Dostupné od:	28. 04. 2014
Umiestnenie:	www.upjs.sk/pracoviska/univerzitna-kniznica/e-publikacia/#pf

ISBN 978-80-8152-139-3