Príručka k použitiu systému VizzA

Alena Košinárová, FMFI UK, 2013

Obsah

1 Úvod

Hneď na začiatku vám chceme poďakovať, že ste sa rozhodli vyskúšať systém VizzA na vizualizáciu programov. VizzA vznikol v rámci diplomovej práce autorky v rokoch 2012-2013 a v rovnakom čase vznikla aj táto príručka pre používateľa.

Primárnym účelom tejto aplikácie je pomôcť pedagógom a študentom pri vyučovaní programovania a algoritmiky modernými metódami, ako aj predviesť nástroj, ktorý poskytuje možnosť vytvárania ilustrácií k textovým publikáciam v súvisiacej oblasti. Ako hlavnú cieľovú skupinu sme brali pedagógov na základných, stredných aj vysokých školách, ale ak aj nespadáte do tejto kategórie, nezúfajte. Našu aplikáciu môžete použiť samozrejme tiež, v plnom rozsahu.

2 Špeciality pre učiteľov

Medzi špeciality určené čisto pre účely vzdelávania, ktoré VizzA inovatívne prináša, patrí Skrytý mód a Kladenie otázok. V stručnosti si spomenieme ich popis a pedagogický význam.

Skrytý mód poskytuje po nahratí programu do aplikácie možnosť zamedzenia zobrazovania zdrojového kódu vizualizovaného programu. Táto možnosť poskytuje nielen novú variantu výučbovej techniky, ale tiež poskytuje viac priestoru pre využitie Kladenia otázok.

Kladenie otázok je bonusová funkcionalita, ktorá pri aplikovaní priradeného príkazu zastaví vizualizáciu a položí používateľovi otázku. Odpoveď používateľa následne aplikácia porovná so zadanou odpoveďou a používateľovi zobrazí výsledok a správnu odpoveď. Týmto spôsobom je možné klásť otázky s absolútne zadanou odpoveďou, alebo tiež otázky na aktuálny stav niektorej premennej. Možností využitia je nespočetne, ale najdôležitejšia je, že dáva vyučujúcemu prostriedky, ako docieliť, že sa konzument vizualizácie samostatne zamyslí nad behom programu.

3 Ako na to

Základný proces tvorby vizualizácie by sa dal zhrnúť do niekoľkých bodov:

- voľba použitých jazykov - na začiatku (aj keď je možné tento krok vykonať aj neskôr, najvhodnejšie je učiniť ho ešte pred začatím programovania) je treba uzrejmiť si, aký programovací a aký vizualizačný jazyk hodláte pri príprave použiť; tieto jazyky je možné nastaviť vpravo hore na paneli nástrojov.

- vloženie programu, ktorý chceme vizualizovať - či už s pomocou importnej funkcie, alebo priameho písania či kopírovania do programového panelu, potrebujete vytvoriť želaný program. V tejto fáze je treba dávať pozor na jemné odchýlky oproti štandardnej syntaxi programovacích jazykov. Priebežne rozpoznané kľúčove slová je možné sledovať v paneli zvýrazneného programu už počas vytvárania.

- vloženie vizualizačných príkazov, alebo upravenie existujúcich príkazov do vizualizačnej formy - na to, aby bol program vizualizovaný, musí obsahovať aj vizualizačné príkazy. Premenné, ktoré majú byť zobrazené, musia byť pri svojom vzniku označené vizualizačným príkazom. Podobne aj príkazy, ktoré majú byť zobrazené, musia byť vizualizačne označené. Okrem úprav (na pôvodných príkazoch) je možné pridať tiež čisto vizualizačné príkazy, ktoré poskytujú doplňujúce možnosti.

- spracovanie programu - s pomocou tlačidla na hornej lište. V prednastavenej sade ikoniek má zobrazenú žiarovku, ktorá je šedivá, pokiaľ aktuálny program v programovom paneli spracovaný nie je, a farebná, pokiaľ už spracovaný je.

- spustenie vizualizácie - s pomocou susedného tlačidla na hornej lište. V prednastavenej sade ikoniek má zobrazenie zelenej šípky, pokiaľ je možné spustiť vizualizáciu. Červený krížik namiesto šipky sa objavuje v prípade, že vizualizácia už beží, alebo ju nie je možné spustiť z iných príčin.

- v prípade, že bola vizualizácia prerušená niektorým z vizualizačných príkazov spôsobujúcich prerušenie - pauza, otázka - tak je vo vizualizácii možné pokračovať stlačením tlačidla pre spustenie programu.

Pozn.: V prípade potreby opätovného spustenia vizualizácie je nutné vizualizáciu znovu spracovať.

3.1 Poznámky k syntaxi jazykov

Vo **všetkých jazykoch** nastali tieto zmeny:

- **definície premenných:** v jednom riadku a príkaze definujte len jednu premennú. Nie je možné spájať definíciu a prvé priradenie hodnoty

- **záporné čísla:** zapisujte, prosím, záporné čísla vo forme 0-10 namiesto -10

- **vynechávanie zátvoriek** alebo begin/end za if alebo cyklami, pokiaľ sa jedná len o jediný vnorený prikaz: nefunguje, treba vždy použiť zátvorky, resp. iné označenie začiatku a konca vnorenej časti programu

- `i++, i-, inc(i), dec(i)`: tieto syntaktické skratky nefungujú, aplikujte, prosím, `i=i+1`, resp. `i=i-1`

- výpis a načítanie **zložených formátov:** nefunguje. Vždy načítavajte v jednom príkaze len jednu premennú. Vždy vypisujte buď reťazec, alebo hodnotu premennej. Nie oboje naraz, a nie zložený reťazcový výraz.
- pre operáciu **zreťazenia** znakov/reťazcov sa používa namiesto znaku `+` znak `.`
- **skracovanie** testovania **pravdivostných hodnôt** na pravdivosť nie je možné, je nutné používať plný zápis `bool==true` namiesto `tool`
- pri **for-cykloch** bude inkrementačný príkaz vždy vykonaný aj na konci poslednej iterácie podprogramu

V **jazyku C** nastali tieto zmeny, a vznikol teda C-typ jazyk:

- for cyklus: v zátvorke sa namiesto bodkočiarok využívajú čiarky; definícia cyklovej premennej sa musí odohrať pred príkazom cyklu, nie v ňom

V **jazyku Pascal** nastali tieto zmeny:

- premenné: je nutné uvádzať `var` na začiatku každej definície premennej.
- repeat cyklus: zmena formátu. Presný formát sa nachádza v tabuľke na konci tejto príručky.
- porovnanie premenných: namiesto `=` využívame jednotné `==`.

4 Nastavovanie vizualizácie

Vzhľad vizualizácie je možné upraviť rôznymi spôsobmi, avšak najdôležitejšou cestou je využitie vizualizačných parametrov. Parametre sa píšu medzi vizualizačný príkaz a telo programového príkazu, pokiaľ nie je povedané inak. Meno a hodnota parametra sú oddelené mriežkou a nesmú obsahovať medzery ani okolo mriežky. Presný formát teda je (pričom pri viacerých parametroch sú jednoducho písané vedľa seba, oddelené od seba medzerou a nezáleží na ich poradí):

```
viz-príkaz meno-parametra#hodnota-parametra telo-príkazu
```

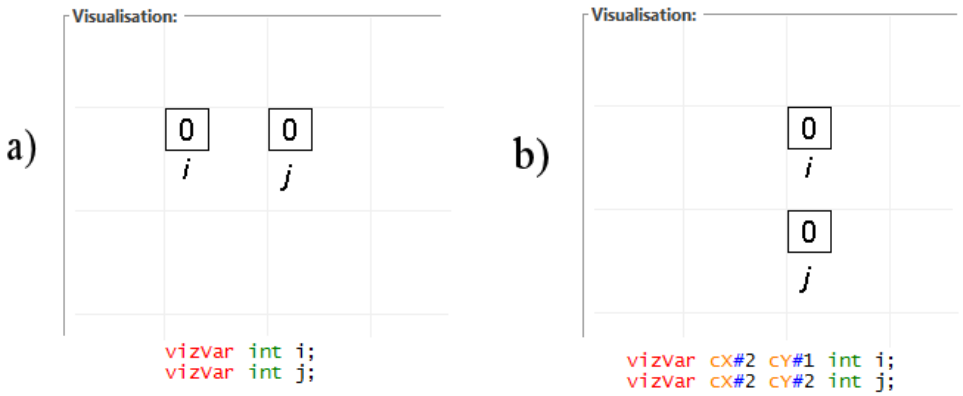
S použitím parametrov dokážeme ovplyvňovať veľa vlastností vizualizácie, a teda sa pozrieme na jednotlivé možnosti samostatne a podrobnejšie. Ešte spomeniem, že všetky parametre majú svoje štandardné označenie, ale niektoré môžu mať okrem toho aj korektné alternatívne označenia podľa použitého vizualizačného jazyka. Kompletný zoznam implementovaných parametrov k jednotlivým príkazom nájdete v tabuľke na konci tejto príručky.

4.1 Umiestňovanie premenných

Premenné, ktoré sa majú vizualizovať, majú svoj zobrazený ekvivalent umiestňovaný do mriežky, ktorej zobrazovanie je možné si zapnúť a vypnúť v paneli nástrojov. Do každého políčka v tejto mriežke je možné umiestniť len jeden prvok, pričom jeden prvok môže zaberať aj viac políčok. Vtedy sa jedná o umiestňovanie jeho ľavej hornej časti.

Pokiaľ nie je nastavené parametrami inak, prvok bude zobrazený v prvom voľnom políčku, pri "čítaní po riadkoch", do ktorého sa zmestí tak, aby sa neprekrýval so žiadnym iným prvkom a tiež netrčal z obrazovky. Pokiaľ sa také miesto nenájde, bude umiestnený do ľavého horného rohu, už bez ohľadu na prekryv. Výskyt tohto javu napovedá, že vizualizovaný panel je zúfalo preplnený a bolo by vhodné zredukovať počet vizualizovaných premenných, alebo ich preusporiadať.

Vždy je však možné určiť umiestnenie premennej podľa vlastného uváženia s pomocou parametrov pre zvislú a vodorovnú polohu - parametre `cX` a `cY`. Využívajú sa pri vytváraní premennej a ich hodnoty majú formu prirodzených čísel, ktoré zodpovedajú poradovému číslu políčka v rámci stĺpcov a riadkov mriežky.



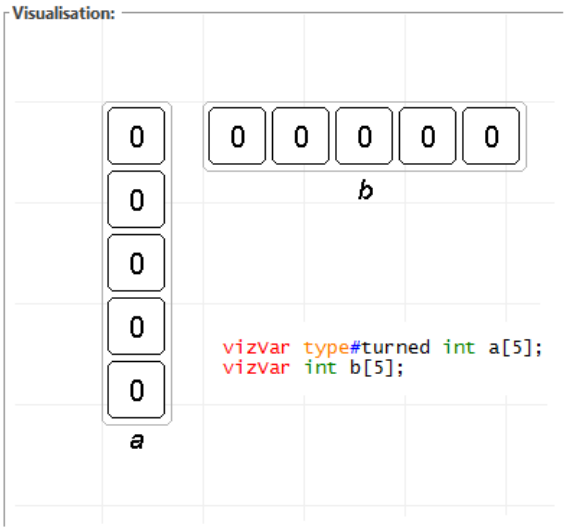
Obr. 1: **Ukážka využitia parametrov pri určovaní pozície.**
a) automaticky umiestnené premenné; b) premenné umiestnené parametrami.

4.2 Typ zobrazenia

S pomocou parametrov je možné nastaviť typ zobrazenia jednotlivých vizualizovaných premenných v prípade, že je k danému typu premennej implementovaných viac ako jedno možné zobrazenie. Meno parametra pre určovanie typu zobrazenia je `type` a uvádza sa hneď pri vytváraní premennej. Typ zobrazenia zodpovedá rozloženiu jed-

notlivých častí premennej voči sebe navzájom. Pokiaľ chcete meniť samotný výzor vizualizácie, prejdite na nasledujúcu časť, ktorou je Dizajn vizualizácie.

Aktuálne sú implementované: jednoduché zobrazenie pre premenné (`simple`), vodorovné (`multiple`) a zvislé (`turned`) zobrazenia pre jednorozmerné polia a tabuľkové zobrazenie pre dvojrozmerné polia (`table`).



Obr. 2: **Ukážka využitia nastavovania typu zobrazenia.**
pole `a` s nastavením zobrazenia na zvislé, pole `b` bez nastavenia zobrazenia a teda s aplikovaným prednastaveným vodorovným zobrazovaním.

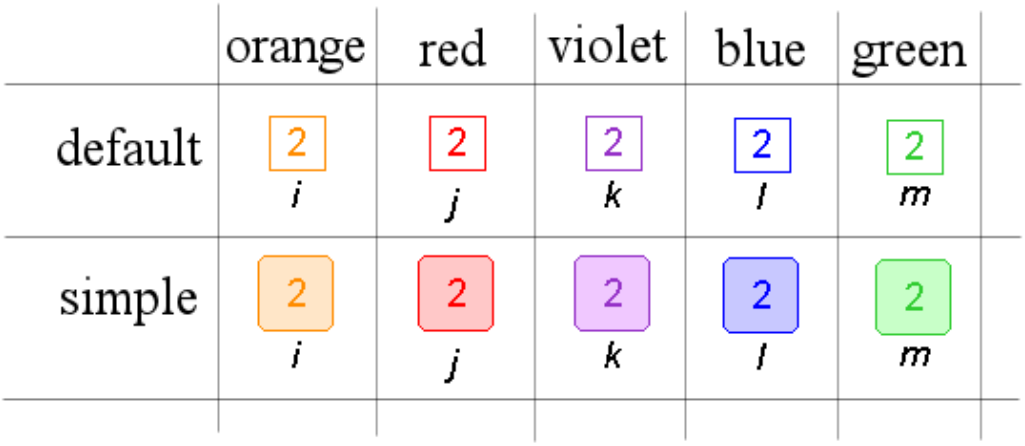
4.3 Dizajn vizualizácie

Okrem základného rozloženia prvkov premennej je nutné určiť, ako presne má premenná pri vykresľovaní vyzerať. Na tento účel slúžia dizajny spoločne s ich parametrami, na ktoré sa pozrieme. S pomocou dizajnu určujeme veľkosť jednotlivých častí vizualizácie, farebné zloženie, typ a veľkosť písma, ako aj mnoho iného. Všetky dizajnové parametre sa majú udávať pri príkaze určenom na vytvorenie premennej.

S pomocou parametra `design` sa dá nastaviť úvodná sada nastavení zhrnutá v niektorom implementovanom dizajne. Takáto sada poskytuje tvary, písma, umiestnenie hodnoty a názvu premennej, ako aj prednastavenú veľkosť a farby.

Vzhľadom na to, že často je možné očakávať, že hodnoty v niektorej premennej budú pri výpise širšie než je prednastavená veľkosť vizualizovaného prvku, je možné upraviť niekoľkonásobne šírku priestoru určeného pre hodnotu premennej. Koeficient rozšírenia je možné nastaviť parametrom `multiplicator`, ktorého hodnota vo forme prirodzeného čísla určuje koľkonásobok prednastavenej šírky má prvok zaberať.

Taktiež je možné samostatne nastaviť farebnú schému, ktorú dizajn používa, aj keď je nutné mať na pamäti, že nie každý dizajn musí využívať striktne len farby z tejto schémy. Farebné schémy sú nastavované s pomocou parametra `color`.



Obr. 3: **Ukážka implementovaných farebných schém.** Aplikované na implementovaných dizajnoch. Označenia riadkov a stĺpcov zodpovedajú hodnotám príslušných parametrov.

4.4 Globálne nastavenia

Keďže očakávame, že od vizualizácií bude aspoň v určitých smeroch žiadaná jednotnosť, pridali sme taktiež možnosť zavedenia globálneho nastavenia prednastavených hodnôt parametrov. Inými slovami, špecifickým príkazom je možné nastaviť vizuálne parametre pre všetky nasledujúce vizualizované premenné, pokiaľ pri nich nebude povedané inak.

Vlastnosť	**Globálny parameter**
Umiestnenie	nedá sa
Typ zobrazenia premennej	`varVisTech`
Typ zobrazenia jednorozmerného poľa	`arrayVisTech`
Typ zobrazenia dvojrozmerného poľa	`duoArrayVisTech`
Dizajn	`design`
Multiplikátor šírky	`multiplicator`
Farebná schéma	`color`

Tabuľka 1: Parametre pre vizualizáciu premenných.

5 Ďalšie funkcie

5.1 Vstup a výstup

Pod vizualizačným panelom sa nachádza miesto pre zadávanie vstupu a čítanie výstupu programu. Tieto dve oblasti fungujú ako náhrada klasického vstupu a výstupu s konzolou. Implementácia využívajúca tieto nástroje je jemne zjednodušená, keďže sa nejedná o vizualizačne kľúčovú vlastnosť. Vstupová konzola však spríjemňuje zadávanie vstupných dát do programu, takže sme sa rozhodli ju predsa len začleniť do VizzA. V implementácii sú medzery považované za oddeľovače vstupných dát, rovnako ako konce riadkov.

5.2 Rozpoznávanie chýb

Vedľa vstupno-výstupných panelov sa nachádza panel chýb, v ktorom sa, v prípade, že VizzA odhalí niektorú chybu, zobrazí popis nájdenej chyby. Pod chybami myslíme chybu v behu vizualizovaného programu. V aktuálnej verzii rozpoznáva VizzA tieto chyby:

- **Index mimo hraníc poľa** - v prípade, že sa snažíme pristupovať do poľa s indexom, ktorý ukazuje mimo zadaných hraníc poľa
- **Premenná nenájdená** - chyba sa objaví, pokiaľ sa pokúšame pracovať s premennou, ktorá v systéme neexistuje
- **Premenná už existuje** - ak sa snažíme vytvoriť viac rovnomenných premenných
- **Chýbajúci vstup** - pokiaľ by sme mali čítať neexistujúci vstup
- **Neinicializovaná premenná** - pokiaľ sa pokúšame pracovať s premennou, do ktorej sme predtým nepriradili hodnotu.
- **Neznámy príkaz** - ako jediná (z implementovaných chýb) sa zobrazuje už počas spracovania programu

5.3 Import a export programov

V rámci pohodlia a praktického použitia je možné zapísať pripravený program aj so vstupnými dátami a ďalšími nastaveniami do exportného textového súboru a ten si nahrať na disk. Podobne je možné z disku získaný súbor spätne vložiť do aplikácie a tým rýchlo a bez problémov posunúť svoj projekt ďalším používateľom. Na účel importu slúži prvá ikonka na paneli nástrojov (v prednastavenej sade je to modrá otvorená

zložka) a na účel exportu slúži štvrté tlačidlo na paneli nástrojov (v prednastavenej sade je to žltá disketa s textovým popisom).

Pozn.: Importný súbor má formát, ktorý je z väčšej časti čitateľný voľným okom, preto pokiaľ ho študenti nemajú vidieť, je nutné ho nahrať do aplikácie a následne odstrániť z ich dosahu.

5.4 Export obrázkov

VizzA ako jednu zo svojich hlavných funkcionalít poskytuje možnosť vytvárania vizualizačných ilustrácií k pedagogickým textom alebo prednáškam. Obrázky je možné exportovať samostatne z vizualizačného panelu, alebo z celej aplikácie. Zatiaľčo export celej aplikácie prebieha zásadne vo formáte png, pre export vizualizačného panelu ponúkame tiež možnosť formátu jpg, v prípade použitia exportného príkazu namiesto tlačidla.

Z tlačidiel slúžia na tento účel druhé a tretie tlačidlo panela, (čo je pri prednastavenej sade ikoniek čierna disketa a žltá disketa s obrázkom), pričom prvé menované vytvára obraz celej aplikácie, zatiaľčo druhé menované nahráva obrázok vizualizačného panelu. Okrem týchto variantov je možné na export využiť tiež príkaz, ktorý vo chvíli, keď sa bude mať vykonať, spustí nahrávanie obrázku aktuálneho stavu vizualizačného panelu. Po úspešnom ukončení nahrávania pokračuje beh programu ďalej. K tomuto príkazu patrí tiež voliteľný parameter `type` resp. `file`, ktorý určuje svojou hodnotou formát výstupného obrázku. Na výber sú hodnoty `jpg` a `png`.

5.5 Používateľské rozhranie

Používateľské rozhranie VizzA sa skladá z viacerých záložiek, z ktorých všetky, okrem poslednej, zodpovedajú rôznym rozloženiam prvkov aplikácie, aby mohol každý používateľ pracovať v takej forme, aká je mu prirodzená. Posledná záložka obsahuje nastavenia týkajúce sa používateľského rozhrania. Takýmito nastaveniami sú napríklad jazyk, alebo tiež povolené záložky. Tieto nastavenia sa pri exporte programov, môžu ale, nemusia tiež exportovať.

6 Tabuľky a schémy

Na nasledujúcich stranách sú uvedené tabuľký s kompletným zoznamom príkazov vizualizačného aj programového charakteru vrátane syntaktického použitia, ako aj kompletný zoznam implementovaných parametrov.

Príkaz	**Slovenčina**	**English**	**Namiesto prog. príkazu**
Pridanie premennej	`vizPremenna`	`vizVar`	áno, ak existuje
Priradenie hodnoty	`vizOperacia`	`vizOperation`	nie, píše sa pred neho
If	`vizIf`	`vizIf`	áno
For	`vizFor`	`vizFor`	áno
While	`vizWhile`	`vizWhile`	áno
Repeat	`vizRepeat`	`vizRepeat`	áno
Načítanie	`vizVstup`	`vizRead`	nie
Výpis	`vizVystup`	`vizWrite`	nie
Prerušenie	`vizVstup`	`vizRead`	čisto vizualizačný
Nahrať obrázok	`vizObrazok`	`vizImage`	čisto vizualizačný
Kladenie otázky	`vizOtazka`	`vizQuestion`	čisto vizualizačný
Globálne nastavenia	`vizNastavenia`	`vizGlobal`	čisto vizualizačný

Tabuľka 2: Vizualizačné príkazy a ich syntax vo VizzA.

Parameter	**Alternatívy**	**Príkaz**	**Hodnoty**
`multiplicator`	`mult` (slovenčina, english)	Premenná Glob. nastavenia	prirodzené čísla
`design`	`dizajn` (slovenčina)	Premenná Glob. nastavenia	`default, simple`
`color`	`farba` (slovenčina)	Premenná Glob. nastavenia	`red, blue, green violet, orange`
`cX, cY`	–	Premenná	prirodzené čísla
`type`	`visual` (english) `typ` (slovenčina)	Premenná Jednorozmerné pole Dvojrozmerné pole	`simple multiple, turned table`
`type`	`file` (všade), `typ` (slovenčina)	Nahratie obrázku	`jpg, png`

`varVisTech`
`arrayVisTech`
`duoArrayVisTech`
	–	Glob. nastavenia	`simple multiple, turned table`

Tabuľka 3: Existujúce vizualizačné parametre.

Príkaz	**C-typ**	**Pascal**

Tvorba premennej	`int i;`
	`boolean b;`
	`string s;`
	`char c;`
	`int a[5];`
	`int a[5][5];`
		`var i : integer;`
		`var b : boolean;`
		`var s : string;`
		`var c: char;`
		`var a: array [1..5] of integer;`
		`var t: array [1..5,1..5] of integer;`

Priradenie	`a[3]=5;`	`a[3]:=5;`

Podmienka	`if (`*`podmienka`*`) {`
	nutne v nových riadkoch...
	`}`
		`if` *`podmienka`* `then begin`
		nutne v nových riadkoch...
		`end;`

Začiatok a koniec tela programu	–	*premenné*
		`begin`
		program...
		`end.`

For cyklus	`int i;`
	`for (i=0,` *`podmienka`*`,`
	`i=i+1) {`
	nutne v nových riadkoch
	`}`
		`var i: integer;`
		`for i:=0 to` *`maximum`* `do`
		`begin`
		nutne v nových riadkoch
		`end;`

While cyklus	`while (`*`podmienka`*`) {`
	nutne v nových riadkoch
	`}`
		`while` *`podmienka`* `do begin`
		nutne v nových riadkoch...
		`end;`

Repeat cyklus	–	`repeat until` *`podmienka`* `begin`
		nutne v nových riadkoch
		`end;`

Načítanie	`readf(%`*`skratkaTypu`*`, i)`	`read(i);`
Výpis	`printf(i);` `printf("reťazec");`	`write(i);` `writeln("reťazec");`
Logické spojky	`&& || !`	`AND OR NOT`
Zreťazenie	`.`	`.`
Porovnanie rovnosti	`==`	`==`
Porovnanie nerovnosti	`!=`	`<>`
Porovnania	`>= <= > <`	`>= <= > <`

Tabuľka 4: Príkazy v programovacích jazykoch a ich syntax vo VizzA. Obsahuje aj zmeny oproti klasickej syntaxi.