Ročníkový projekt z programovania, 1998/99

Martin KRÁLIK

Algoritmy a dátové štruktúry '99

1 Užívateľská príručka

1.1 O programe

Program má názov *Algoritmy a dátové štruktúry* a tento vôbec nie je samoúčelný. Program formou učebnice s prvkami interaktivity jednoducho vysvetľuje a vizualizuje algoritmy triedenia, vyhľadávania, popisuje jednoduché aj pokročilé dátové štruktúry a niektoré z nich umožňuje užívateľovi aj vytvárať, či modifikovať. Program je členený do týchto logických častí:

1. **Spájané zoznamy -** obsahuje popis všetkých typov spájaných zoznamov, ktoré sa reálne v praxi používajú, teda jednosmerných, obojsmerných aj cyklických zoznamov s nárazníkom. V tejto časti sú vizualizované všetky základné operácie nad zoznamami, teda vytváranie zoznamu, vkladanie prvkov do zoznamu aj vymazávanie prvkov zo zoznamu. Vizualizácia je plne interaktívna v tom zmysle, že užívateľ má možnosť si zoznamy od prvého po posledný prvok vytvoriť sám za pomoci myši

2. **Binárne stromy -** v tejto časti je popísaná táto veľmi často používaná dátová štruktúra, sú tu uvedené najpoužívanejšie algoritmy na prechod binármymi stromami, na ktorých základe je založená väčšina ostatných algoritmov určených pre prácu s binárnymi stromami (konkrétne ide o algoritmy *Preorder*, *Inorder* a *Postorder*, všetky tri sú vizualizované)

3. **N-árne stromy -** vysvetlenie abstraktného pojmu n-árny strom, je tu uvedená reprezentácia n-árnych stromov pomocou binárnych stromov pre prípad neohraničeného n

4. **Grafy -** pokročilá dátová štruktúra, sú tu uvedené základné pojmy z teórie grafov, reprezentácia grafov v počítači, ako ukážkový algoritmus je uvedené nájdenie kostry grafu, tu je demonštrované prehľadávanie grafu do šírky, algoritmus je opäť plne interaktívny, graf zadáva pomocou myši užívateľ

5. **Triedenie -** popisuje najpoužívanejšie algoritmy triedenia dát, niektoré z nich sú vizualizované, každý algoritmus je pomerne podrobne popísaný

6. **Vyhľadávanie -** obsahuje tri základné časti
   (a) *BST* - binárne vyhľadávacie stromy - ich popis, výpis algoritmom *Inorder* (je utriedený), vytváranie vlastných *BST* užívateľom
   (b) *Hašovanie* - popis tejto metódy ukladania a vyhľadávania dát
   (c) *B-stromy* - definícia, popis
7. **Halda -** popis dátovej štruktúry, triedenie *Heapsort* ako aplikácia haldy

Takmer ku všetkým algoritmom a dátovým štruktúram sú tu uvedené aj ich implementácie v jazyku PASCAL (resp. sú to akési pseudoalgoritmy s malou úrovňou abstrakcie, ktoré by však nemal byť problém pochopiť).

1.2 Ovládanie programu

Je vcelku intuitívne. Program pre svoju prácu potrebuje myš (bez prítomnosti ovládača tohto zariadenia v pamäti sa nespustí), pomocou nej je realizovaný pohyb v menu a je potrebná aj pri vizualizácii niektorých algoritmov. V programe je takmer v každom momente (až na pár výnimiek, kde to nie je vhodné) prístupná klávesa *Esc*, pomocou ktorej sa vrátite o úroveň vyššie (program je totiž štrukturovaný do užívateľských submenu, aby sa uľahčila orientácia v ňom). Ak si náhodou nebudete vedieť rady a už už budete v duchu preklínať autora, skúste si všimnúť stavový riadok, ktorý vám vždy, keď je to potrebné, povie, akú klávesu treba stlačiť, keď chcete vykonať to, či ono.

1.3 Systémové požiadavky

Program je vcelku nenáročný, berie ohľad na to, že ak by bol náhodou v budúcnosti využívaný na nejakých tých stredných školách, aby tam mohol na dostupnom vybavení pracovať (uznajme, že toto je nie vždy najnovšie)

- PC-AT 386
- VGA grafická karta
- myš (je nevyhnutnou súčasťou vybavenia)
- MS-DOS 6.2

2 Programátorská dokumentácia

2.1 Motivácia

K téme môjho projektu ma motivovala moja programátorská mladosť, keď som kdesi na strednej škole začínal s PASCAL-om a bol som doslova hladný po informáciách o "dobrom" programovaní. Ich dostupnosť sa však prudko blížila k bodu 0, a tak mi nezostalo nič iné, len improvizovať. Keď si na to niekedy spomeniem.. No, musel som si vymýšľať algoritmy takmer na všetko. Keď som potreboval utriediť pole, to najlepšie, čo ma vtedy napadlo, bol BUBBLESORT, aj to som vtedy nevedel, že sa to tak volá. Tak som si povedal.. Prečo by sa aj iní museli takto trápiť, keď sa im dá pomôcť.. A tak vo mne skrsla myšlienka napísať tento program.. Ak sa nikdy nevyužije, nebudem sklamaný. Venujem ho mojej programátorskej mladosti a útrapám, ktoré s ňou boli spojené.

2.2 Vylepšenia (veci, ktoré nevidno)

Ide o veci, ktoré dnešný *Win-oriented* užívateľ nevidí, pretože ich pokladá za samozrejmosť, no ony až také samozrejmé nie sú (hlavne v TP, kde ich treba pracne programovať). V prípade môjho projektu som si musel naeditovať vlastné myšacie kurzory. Na ich editáciu som si navrhol vlastnú utilitku, ktorú som si, žiaľ, nechtiac nenávratne zmazal a myslím, že nebolo potrebné ju znova programovať. Ďalej bolo potrebné navrhnúť vlastný formát súboru na ukladanie textovej i grafickej informácie súčasne. Najskôr som rozmýšľal o hypertexte, no túto myšlienku som razom s jej rozpracovaním zamietol. Vytvoril som si teda vlastný formát súboru a konvertor textového súboru do tohto formátu (uložený v súbore *convert.pas*).

2.3 Problémy pri implementácii

2.3.1 Technické problémy

Najväčším problémom pri implementácii programu bola neskúsenosť s projektami takéhoto rozmeru. Program má v konečnej fáze zhruba 6500 riadkov a priznám sa, že je to takmer trojnásobne viac, ako mal môj doteraz najväčší projekt. Zo začiatku som volil alternatívu, kedy som vlastne, až na výnimky, celý zdrojový text udržoval v jednom súbore *projekt.pas*. Samozrejme riadky sa kopili a kôli prehľadnosti som sa musel rozhodnúť pre jednu z dvoch ponúkaných alternatív:

1. rozdeliť text na *unity*

2. rozdeliť text pomocou direktívy $I na viacero súborov

Ja som si v domnení lepšej voľby vybral druhú možnosť, ktorá sa ukázala byť po čase horšou. Nastali problémy s veľkosťou kódu, a keďže som preklad robil v *real* móde, bol som nútený text rozdeliť na *unity*. Bolo to dosť prácne, pretože niektoré kľúčové procedúry sa prestali chovať korektne, no problém bol riešiteľný. Iné technické problémy sa nevyskytli.

2.3.2 Programátorské problémy

Boli. Na prvý pohľad jednoduché úlohy sa stali takými tvrdými orieškami, že som v istých momentoch nevedel, ako ďalej (našťastie ich nebolo mnoho)

1. mal som problémy s vizualizáciou obojsmerne spájaného zoznamu, síce som si vytvoril knižnicu pre prácu so zoznamami, dokonca som ju rozšíril o obojsmerne spájané zoznamy, no nie a nie ich korektne zobrazovať.. Povedal som si, že si prácu mierne zjednoduším, a tak som sa dopustil malého "podvodu": obojsmerne spájaný zoznam som v pamäti implementoval ako zoznam jednosmerný, no vizualizoval som ho obojsmerne[1]

2. druhý závažný problém bol s vizualizáciou triedenia, vyriešil som ho dosť veľkým okresaním tohto prvku programu, taktiež si nemyslím, že to bol až taký veľký prečin, pretože keď som si pozeral staré projekty, jeden z nich bol venovaný len a len triedeniu a ja sa zaoberám obzorom oveľa širším. Tým som však nechcel povedať, že si v budúcnosti túto vizualizáciu (aj keď možno len pre seba) nedopracujem

2.4 Záver

Ak bude v budúcnosti čas a príležitosť, rád by som v práci na tomto projekte pokračoval. Niektoré algoritmy, ktoré v tejto verzii ešte neboli vizualizované, by si to totiž určite zaslúžili. Ide hlavne o algoritmy triedenia *Heapsort, Mergesort* a *Quicksort*, doplnenie nových operácii pre *BST*, vizualizácia *B-stromov.*

[1] toto je jediné miesto v mojom projekte, kde som sa takéhoto malého programátorského klamstva dopustil, nemyslím si však, že je to na škodu veci, pretože mi to uľahčilo prácu a užívateľ si to nevšimne, aj keby sa veľmi snažil