

Podręcznik użytkownika programu VIP

Michał Adamaszek

1 listopada 2006

Spis treści

| | | |
|----------|------------------------------------|----------|
| 1 | Wprowadzenie | 1 |
| 2 | Interfejs użytkownika | 1 |
| 2.1 | Okno | 1 |
| 2.2 | Uruchamianie | 2 |
| 3 | Możliwości języka | 3 |
| 3.1 | Podstawy | 3 |
| 3.2 | Zmienne wejściowe | 3 |
| 3.3 | Dodatkowe funkcje | 3 |
| 3.4 | Typy specjalne | 4 |
| 3.5 | Obsługa tablic | 4 |
| 3.6 | Kontrola obsługi pamięci | 4 |
| 4 | Możliwości wizualizacji | 5 |
| 4.1 | Podstawy | 5 |
| 4.2 | Widoczność zmiennych | 5 |
| 4.3 | Indeksy tablic | 5 |
| 4.4 | Rekurencja | 6 |
| 4.5 | Typy specjalne | 6 |
| 5 | Uwagi | 6 |

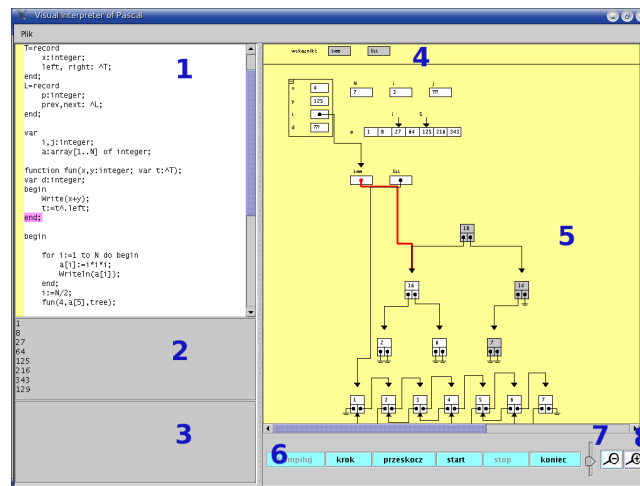
1 Wprowadzenie

VIP (Visual Interpreter of Pascal) jest interpreterem znacznego podzbioru Pascala z pewnymi dodatkami, który wizualizuje działanie programu (zawartość pamięci, stosu) na ekranie. Został napisany w języku Java i jest dostępny jako applet oraz wersja stand-alone.

2 Interfejs użytkownika

2.1 Okno

Okno VIP-a zawiera następujące elementy:



Rysunek 1: Typowy podział okna VIP-a

- 1) — kod wykonywanego programu. Bieżąca linia jest podświetlona. Po lewej obszar do zaznaczania punktów zatrzymania (break-point) w programie.
 - 2) — konsola, na którą kierowane jest wyjście z programu.
 - 3) — konsola na komunikaty interpretera (o błędach w czasie kompilacji, wykonania i inne dodatkowe informacje).
 - 4) — panel przycisków odpowiadających zmiennym w programie. Kliknięcie odpowiedniego przycisku powoduje włączenie/wyłączenie widoczności danej zmiennej.
 - 5) — główny panel, na którym wyświetlana jest zawartość pamięci w czasie działania programu.
 - 6) — przyciski sterujące wykonywaniem programu: kompilacja, zatrzymywanie, uruchamianie itp.
 - 7) — suwak służący do regulacji szybkości działania interpretera.
 - 8) — przyciski skalujące grafikę
- menu Plik — (tylko wersja stand-alone) menu pozwalające wczytywać/zapisywać programy z/do plików.

2.2 Uruchamianie

VIP działa w cyklu edycja — kompilacja — uruchomienie. W fazie edycji (i tylko wtedy) można dokonywać zmian w treści programu i wczytywać/zapisywać z/do pliku. Pozostałe elementy interfejsu pozostają nieczynne.

Przycisk *Kompiluj* uruchamia fazę kompilacji, w czasie której pojawienie się błędu powoduje powrót do fazy edycji z odpowiednią informacją dla użytkownika. Bezpośrednio po udanej kompilacji następuje wczytanie zmiennych

wejściowych (patrz 3.2) i inicjalizacja wszystkich zmiennych — czynności te zaliczamy już do fazy uruchamiania.

Dalszy ciąg fazy uruchamiania jest inicjalizowany przez użytkownika przyciskami *Krok*, *Przeskocz*, *Start*. Interpretację programu można wstrzymać przyciskiem *Stop*. Przycisk *Koniec* służy do porzucenia programu i powrotu do trybu edycji (gdy wykonywany program zakończy się sam, poprawnie bądź nie, interpreter nie powraca sam do trybu edycji aby umożliwić analizę końcowych wartości zmiennych).

W fazie uruchomienia, w dowolnym momencie, można wstawiać i usuwać punkty zatrzymania (breakpoint) programu. Breakpointy ustawione w liniach, w których nie ma wykonywalnej instrukcji są ignorowane. Uruchomienie przyciskiem *przeskocz* także ignoruje breakpointy.

3 Możliwości języka

Ten rozdział opisuje język programów VIP-a. Nie wszystkie konstrukcje językowe podlegają wizualizacji — te kwestie poruszone są w następnym rozdziale. Z kolei formalna składnia podana jest w osobnym załączniku do tego podręcznika.

3.1 Podstawy

VIP opiera się na podzbiorze Pascala zawierającym: liczby całkowite i rzeczywiste, zmienne logiczne, zmienne proste, tablice, rekordy (bez wariantów), wskaźniki, typy użytkownika, funkcje i procedury (nie zagnieżdżone) z parametrami przekazywanymi przez zmienną i wartość, konstrukcje językowe: pętle *while* i *for*, instrukcję warunkową i przypisania, instrukcję wyjścia (*Write*). W tym zakresie powinien pokrywać się z Pascallem.

3.2 Zmienne wejściowe

Program może mieć, analogicznie jak procedura, zmienne wejściowe. Przykład:

```
program binSearch(x,N: integer; a: array[1..N] of integer);
```

Wartości zmiennych wejściowych wczytywane są od użytkownika po zakończeniu kompilacji. Mogą one być typu prostego, typu jednowymiarowa tablica elementów typu prostego lub specjalnych typów opisanych w 3.4 (w przypadku wykorzystania zmiennej innego typu pozostanie ona niezainicjalizowana). Wartość takiej zmiennej można wpisać ręcznie lub skorzystać z predefiniowanych sposobów inicjalizacji: losowo, rosnąco/malejaco (dla tablic) itp. Można też poprosić o inicjalne wartościowanie takiej zmiennej tak samo jak przy poprzednim uruchomieniu programu.

3.3 Dodatkowe funkcje

Wbudowane są funkcje `sqrt(real):real`, `max(...)` i `min(...)` dla dowolnej liczby parametrów oraz instrukcja `zamien(a,b)` o następującej semantyce:

```
zamien(a,b) == x:=a; y:=b; b:=x; a:=y;
```

3.4 Typy specjalne

Wizualizacja niektórych struktur danych jest specjalizowana, tak aby jak najlepiej przedstawić typowe dla tych struktur operacje. Są to: listy, listy dwukierunkowe i drzewa binarne. Struktury tego typu można w całości wczytywać jako zmienne wejściowe poprzez wskaźnik do początku struktury. Przykład:

```
program treeInsert(x: integer; t: ^Tree);
```

(typ `Tree` musi być zdefiniowany w programie). Interfejs wspomaga inicjalizację takich złożonych zmiennych, w tym np. określanie struktury drzewa czy długości listy.

Typy rekordowe, które mają być traktowane jako specjalne, są rozpoznawane po odpowiedniej strukturze. Początek rekordu zajmują dowolne pola własne a następnie:

- dla listy ostatnie pole nosi nazwę `next` lub `nast`
- dla listy dwa ostatnie pola noszą nazwy `prev` i `next` lub `poprz` i `nast`
- dla drzewa binarnego dwa ostatnie pola to `left` i `right` lub `lewy` i `prawy`

Inicjalizować można te pola rekordów, które są typów prostych.

3.5 Obsługa tablic

Zakresy tablic mogą być wyrażeniami (jak w 3.2), pod warunkiem, że są dostatecznie proste tzn. odwołują się jedynie do zmiennych i operacji arytmetycznych (nie odwołują się do funkcji, pól rekordów ani tablic). Wartość wyrażenia opisującego zakres będzie obliczana w chwili tworzenia tablicy. W razie niepowodzenia następuje błąd czasu wykonania.

Instrukcja przypisania dla tablic jest semantycznie poprawna jeśli obie mają ten sam typ i wymiar, ale niekoniecznie długość w każdym wymiarze. W razie różnych długości nastąpi obcięcie lub wypełnienie zmiennymi niezainicjalizowanymi.

Deklaracja `array[1..N] of array[1..M] of integer` nie jest równoważna `array[1..N,1..M] of integer`. Tę drugą VIP potraktuje jako sygnał, że dwuwymiarowa tablica ma być wizualizowana, tymczasem w pierwszym przypadku nie będzie wizualizacji bo typ danych w zewnętrznej tablicy jest zbyt skomplikowany.

3.6 Kontrola obsługi pamięci

Na początku wszystkie zmienne (oprócz wejściowych, a dla funkcji oprócz parametrów funkcji) są niezainicjalizowane. Próba czytania z niezainicjalizowanej zmiennej zakończy się błędem. W szczególności obejmuje to następujące sytuacje: przypisanie na zmienną typu złożonego, gdy któraś składowa czytanej zmiennej jest niezainicjalizowana; wywołanie funkcji/procedury z niezainicjalizowanym parametrem przekazywanym przez wartość.

Zmienna raz zainicjalizowana już na zawsze taka pozostanie, poza jedynym wyjątkiem: wywołanie `dispose(p)` powoduje, że wskaźnik `p` staje się niezainicjalizowany, podobnie jak wszystkie wskaźniki, które wskazywały na tą samą

zmienną co p. Wywołanie `dispose` na wskaźniku wskazującym na zmienną statyczną nic nie robi.

Zmienne, do których utracono wszystkie odwołania (zombie) są wykrywane i zaznaczane poprzez umieszczenie na szarym tle.

Błąd powoduje także odwołanie do indeksu spoza zakresu tablicy.

4 Możliwości wizualizacji

4.1 Podstawy

W odpowiednim panelu pokazywana jest zajętość pamięci: zmienne i ich wartości. Oczywiście nie da się dobrze wizualizować zmiennych o dowolnym stopniu złożoności (tablica rekordów rekordów tablic wskaźników itp.). Obowiązują następujące zasady:

- zmienne proste i wskaźnikowe są zawsze widoczne
- rekordy są widoczne, w nich widoczne są wartości zmiennych typów prostych i wskaźnikowych zaś pola rekordów zawierające bardziej skomplikowane obiekty są „puste”
- widoczne są jedno- i dwu-wymiarowe tablice zmiennych prostych

4.2 Widoczność zmiennych

Wskaźniki są widoczne jako strzałki. Kliknięcie w pole wskaźnika (początek strzałki) powoduje jej podświetlenie, co często ułatwia odnalezienie obiektu docelowego. Wskaźniki, które wskazują w obiekty niewidoczne (głębsze w hierarchii zagnieżdżenia zmiennych) są rysowane do ich widocznych przodków.

Widoczność zmiennych można włączać/wyłączać klikając w przyciski związane z tymi zmiennymi u góry ekranu. Jeżeli wartość zmiennej nie mieści się w przeznaczonym na nią okienku, kliknięcie w to okienko spowoduje wyświetlenie pełnej wartości zmiennej w oknie komunikatów interpretera.

Przyciskami z „lupą” można zmniejszać i zwiększać skalę obiektów widocznych w oknie ze zmiennymi. Jest to alternatywa dla przewijania okna ze zmiennymi jeżeli jest ich dużo.

4.3 Indeksy tablic

Indeksy występujące w odwołaniach do widocznych tablic są zaznaczane przy odpowiedniej tablicy strzałką nad komórką na którą wskazują. Indeks zaczyna być widoczny bezpośrednio przed wywołaniem linii w której występuje i pozostaje widoczny do końca działania programu lub procedury w której został użyty; jego położenie jest przy tym cały czas aktualizowane, tzn. wartość wyrażenia indeksującego jest przeliczana ilekroć zmieniają się wartości użytych w nim zmiennych. Z tego względu indeksy zawierające wywołania funkcji nie są wizualizowane.

Ograniczone miejsce powoduje, że jednocześnie widać najwyżej 3 indeksy do jednej komórki tablicy. Indeks może też wskazywać bezpośrednio przed lub za tablicą, jeżeli wartość wyrażenia indeksowego nie mieści się w zakresie tablicy (nie jest to oczywiście błąd dopóki nie będzie odwołania do takiego indeksu).

4.4 Rekurencja

Jeżeli program zawiera funkcje bądź procedury można oglądać stos. Ze względu na ograniczone miejsce, na stosie wizualizowane są tylko zmienne proste i wskaźnikowe. Parametry przekazane przez zmienną są pokazane jako wskaźnik do tej zmiennej. Na stosie znajduje się także wartość zwracana z funkcji.

Każdą ramkę stosu można pokazywać lub chować w każdej chwili, klikając przycisk $+$ w rogu ramki. Kliknięcie w ramkę powoduje podświetlenie miejsca w kodzie programu w którym nastąpiło wywołanie funkcji, które spowodowało stworzenie tej ramki.

4.5 Typy specjalne

Dla typów specjalnych (patrz 3.4) ich kolejne pola są szeregowane w sposób naturalny dla danego typu danych, np. w strukturę drzewiastą dla drzew (co nie zachodzi dla dowolnych rekordów ze wskaźnikami gdyż nie wiadomo jakie wskaźniki za co miałyby odpowiadać i nie ma naturalnych sposobów porządkowania). Operacje na wskaźnikach wyznaczających strukturę przekładają się na płynne operacje na widocznych obiektach (np. łączenie list, wstawianie elementu itp.). Wskaźniki do takich obiektów realizowane są podobnie jak indeksy do tablic, tzn. strzałką z nazwą wskaźnika obok obiektu.

5 Uwagi

VIP korzysta z rozszerzonej na własne potrzeby wersji generatora parserów Coco/R: (<http://www.ssw.uni-linz.ac.at/Research/Projects/Coco/>) autorstwa H.Mossenbocka, M.Loberbauera i A.Wossa z University of Linz.

Załącznikiem do tego dokumentu jest opis składni języka interpretowanego przez VIP.