Język YAPL - Specyfikacja

Paweł Piecyk
22.05.2012

Język **YAPL** (skrót od *Yet Another Programming Language*) jest językiem interpretowanym i powstał jako projekt zaliczeniowy *Języki formalne i automaty*, na Wydziale Fizyki i Informatyki Stosowanej, AGH Kraków.

Typy danych

Język YAPL jest językiem dynamicznie typowanym, zatem typ danej zmiennej zależy tylko i wyłącznie od wartości, która aktualnie się w niej znajduje. Możliwe wartości to:

- 32 bitowa zmienna liczbowa (integer). (od −2147483648 do 2147483647)
- Liczba zmiennoprzecinkowa (float). Ten typ danych pozwala na przechowywanie 32 bitowych liczb zmiennoprzecinkowych pojenynczej precyzji w standardzie IEEE 754.
- Łańcuch znaków (string).

Operatory

Język definiuje następujące operatory:

Operator przypisania <-
Służy do przypisywania wartości zmiennym a także przekazywania parametrów do wywoływanych procedur. Przykład: set zmienna <- 4;

Operatory logiczne (== < > <= >= !=)
Służą do porównywania wartości zmiennych. Zwracają zmienną typu **integer**.

Operatory arytmetyczne (+ - * /)
Służą do tworzenia wyrażeń arytmetycznych. Operatory mnożenia i dzielenia (* i /) mają wyższy priorytet niż + i -. Priorytet można zmieniać stosując nawiasy okrągłe.

Instrukcje

Tworzenie zmiennych

Zmienne tworzymy używając instrukcji **set.** Możemy również jednocześnie zdefiniować ich wartość z użyciem operatora przypisania (<-). **Uwaga, wszystkie zmienne mają zasięg globalny w zakresie całego programu!** Przykład kodu:

```
set nazwa_zmiennej <- 99;
```

Wypisywanie na ekran

Wypisywanie na ekran odbywa się za pomocą dostarczonej instrukcji **dspl**, która przyjmuje jako parametr zmienną. Przykład kodu:

```
set a <- "Hello World";
dspl a;
```

Instrukcja warunkowa

Instrukcja warunkowa w YAPL ma postać:

```
check warunek:
@b
	#instrukcje wykonywane gdy warunek prawdziwy
@e
nottrue:
@b
	#instrukcje wykonywane gdy warunek nieprawdziwy
@e
```

Pętle

W YAPL dostępne są dwa typy pętli (właściwie jeden, ale działający na dwa sposoby). Prosta pętla podobna do pętli for:

```
loop warunek_zakończenia ^ instrukcja_np_inkrementacji:
@b
	#instrukcje
@e
```

Możemy pominąć część po znaku ^, otrzymamy wtedy pętlę działającą jak pętla **while**.

Komentarze

Obsługiwane są jedynie komentarze jednolinijkowe – komentarz zaczyna się od znaku #

Tworzenie procedur

Procedury tworzymy używając słowa kluczowego **def**. Składnia jak poniżej:

```
def suma : liczba1, liczba2
@b
set s <- liczba1+liczba2;
dspl s;
@e
```

Procedurę wywołujemy w następujący sposób (przykład dla powyższej):

```
!suma <- 4,6;
```

Procedura główna

Główna część programu zawarta jest w procedurze **start_point**, która nie przyjmuje żadnych parametrów.

Przykładowy program

```
set x <- 0;
def suma: a, b
@b
      x <- a+b;
@e

def start_point:
@b
      set liczba1 <- 10;
      set liczba2 <- 20;
      !suma <- liczba1, liczba2;
      dspl "Suma wynosi:";
      dspl x;
@e
```