AKADEMIA PODLASKA

WYDZIAŁ NAUK ŚCISŁYCH
INSTYTUT INFORMATYKI

Interpreter języka Pascal zaimplementowany w Java

Waldemar Bartyna
Marek Faderewski
Łukasz Fedorczyk
Wojciech Iwanowski

Siedlce 2005 r.

Spis treści

1. Cel projektu.

Celem naszego projektu było stworzenie aplikacji, której zadaniem jest interpretowanie kodu napisanego w języku Pascal. Aplikacja została napisana w języku Java. Wykonanie programu przebiega przez następujące etapy:

- **Skanowanie** – wyszukuje leksemy w analizowanym programie
- **Walidacja** – sprawdza poprawność składni kodu (syntaktyka)
- **Interpretacja** – wykonuje kod programu

2. Założenia projektowe.

- Obsługiwane typy danych:
  - Integer
  - Real
  - Boolean
- Obsługiwane instrukcje
  - Instrukcje przypisania
  - Instrukcje warunkowe
  - Instrukcja pętli *while* (tego typu instrukcja pozwala na zaimplementowanie wszystkich rodzajów pętli)
- Nie ma ograniczeń co do zagnieżdżania instrukcji

3. Wymagania sprzętowe i programowe.

- sprzętowe

- Procesor 100 MHz x86
- RAM 16 MB

- Programowe
  - System operacyjny z maszyną wirtualną Java

4. Specyfikacja BNF.

```
<cyfra> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<liczba> ::= [„-„] <cyfra>{<cyfra>}

<liczba_dziesietna> ::= [„-„] <liczba>”.”<liczba>

<litera> ::= a | b | c | d | e | f | g | h | i | j |
             k | l | m | n | o | p | q | r | s | t |
             u | v | w | x | y | z | A | B | C | D |
             E | F | G | H | I | J | K| L | M | N |
             O |P | Q | R | S |  T | U |V | W | X |
             Y | Z

<slowo> ::= <litera> {<litera> | <cyfra>}

<znak_arytmetyczny> ::= ”+” | „-” | „*” | „/”

<operator_przypisania> ::= „:=”

<operator_porownania> ::=  „>=” | „<=” | „<>” | „=” |
                           ”>” | ”<”

<operator_logiczny> ::= „not” | „and” | „or”

<typ_danych> ::= “integer” | “real” | “Boolean”

<t_program> ::= “program”

<t_var> ::= “var”

<t_begin> ::= “begin”

<t_end> ::= “end”

<koniec_programu> ::= “end.”
```

```
<t_if> ::= „if”

<t_then> ::= „then”

<t_else> ::= “else”

<t_while> ::= “while”

<t_do> ::= “do”

<slowo_kluczowe> ::= <typ_danych> | <t_program> |
              <t_var> | <t_begin> | <t_end> |
              <koniec_programu> | <t_if> | <t_else> |
              <t_while> | <t_do>


<nazwa_zmiennej> ::=  <slowo> except for
<slowo_kluczowe>

<deklaracja_zmiennej> ::= <nazwa_zmiennej> { „,”
            <nazwa_zmiennej> } „:”  <typ_danych> „;”

<blok_deklaracji_zmiennych> ::=  <t_var>
    <deklaracja_zmiennej>{ <deklaracja_zmiennej> }

<wyrazenie_arytmetyczne> ::= <liczba> |
        <nazwa_zmiennej>   <operator_arytmetyczny>
        <liczba> | <nazwa_zmiennej>

<wyrazenie_arytmetyczne> ::= <wyrazenie_arytmetyczne>
            <operator_arytmetyczny> <liczba> |
            <nazwa_zmiennej>

<wyrazenie_arytmetyczne> ::=
                „(„ <wyrazenie_arytmetyczne> „)”

<wyrazenie_logiczne> ::= „true” | „false” |
                         <wyrazenie_logiczne>

<wyrazenie_logiczne> ::= <liczba> | <nazwa_zmiennej>
                <operator_porownania> <liczba> |
                <nazwa_zmiennej>
```

```
<wyrazenie_logiczne> ::=  <wyrazenie_logiczne>
            <operator_logiczny> <wyrazenie_logiczne>

<wyrazenie_logiczne> ::= „(„<wyrazenie_logiczne> „)”

<instrukcja_przypisania> ::= <nazwa_zmiennej> ”:=”
            <liczba> | <nazwa_zmiennej>  |
            <wyrazenie_arytmetyczne> |
            <wyrazenie_logiczne> „;”

<instrukcja> ::= <instrukcja_przypisania> |
        <instrukcja_pętli> | <instrukcja_warunkowa>

<instrukcja> ::= <t_begin> {  <instrukcja> } <t_end>

<instrukcja_pętli> ::= <t_while> <wyrażenie_logiczne>
                          <t_do> <instrukcja>

<instrukcja_warunkowa> ::= <t_if>
        <wyrażenie_logiczne> <t_then> <instrukcja>
        [ <t_elese> <instrukcja> ]

<blok_programu> ::= <t_program> <nazwa_zmiennej>
        [ <blok_deklaracji_zmiennych> ] <t_begin>
        <instrukcja> <koniec_programu>
```

4. Uzupełnienie własne

Priorytety w operacjach logicznych:

- operatory porównania,
- NOT,
- AND,
- OR.

Program pozwala na:

- dowolne użycie bloków, zagnieżdżeń,
- dowolne operacje logiczne i arytmetyczne, ale nie jednocześnie, tzn. nie można stosować operacji arytmetycznych w warunkach logicznych,

Program nie obsługuje funkcji:

- np.: potęga, moduł (abs), sinus itp.

Znak „–„ przed zmienna nie powoduje zmiany jej znaku. Konieczne jest napisanie zamiast „–x ” instrukcji „-1*x”

Zwalczono jedną z największych złośliwości Pascala: od tej pory można stawiać średnik przed elsem ! ☺

5. Instrukcja obsługi.

Aplikacją w której można uruchomić interpreter jest Jcreator. Aby wystartować proces interpretacji trzeba uruchomić interpreter (interpreter.java). Ścieżkę do pliku z kodem Pascala (domyślnie znajduje się on w katalogu bieżącym i nazywa się pascal.txt) podaje się jako wartość zmiennej w pliku interpreter.java. Po jej wpisaniu należy skompilować plik.