Implementacja interpretera języka programowania ogólnego przeznaczenia z wbudowanym typem danych przedstawiającym datę.

Daniel Górniak

2 listopada 2021

1 Zasady działania języka.

- obsługuje liczby całkowite i ułamki
  - obsługuje operacje matematyczne o różnym priorytecie wykonania
  - obsługuje operacje logicznie i porównania o różnym priorytecie wykonania
- obsługuje typ znakowy
  - obsługuje konkatenacje napisów tylko z innymi napisami
  - może zawierać dowolne znaki, też wyróżnik napisu (””)
- obsługa operatorów porównania dwóch dat
- w języku tym będzie można pisać komentarze
- obsługuje tworzenie zmiennych, przypisywanie do nich wartości oraz je odczytywać
  - typowanie statyczne
  - typowanie słabe dla int i float
  - mutowalne
  - zmienne będą miały zakresy lokalne
- obsługiwana będzie pętla warunkowa if else
- obsługiwana będzie bazowa pętla While
- obsługiwana będzie możliwość wołania i definiowania własnych funkcji ze zmiennymi lokalnymi, gdzie argumenty będą przekazywane przez wartość
- funkcja wbudowana print przyjmująca jeden argument
- obsługa rekursywnych wywołań funkcji
- obsługa błędów blokiem try exept
- wbudowana funkcja print przyjmująca jeden argument typu napis
- zmienna boolowska false zastąpiona jest zmienną int i float o wielkości 0, inne dają wynik true

2 Struktura projektu.

- projekt zostanie napisany w c++
- będzie to aplikacja okienkowa do której będzie się podawało skrypt, który będzie poddany interpretacji
- testowanie za pomocą testów jednostkowych z użyciem biblioteki Boost
- Moduły:
  - moduł analizatora leksykalnego, czyta ciąg znaków i tworzy kolejne tokeny po prośbie analizatora składniowego; wykrywa nieprawidłowe tokeny i sygnalizuje to modułowi obsługi błędów
  - moduł analizatora składniowego, prosi analizator leksykalny i tworzy z nich drzewa rozbioru, wykrywa błędy składniowe; wykrywa nieprawidłową składnię kolejnych tokenów i sygnalizuje to modułowi obsługi błędów
  - moduł analizatora semantycznego, sprawdzając utworzone drzewa rozbioru sprawdza czy mają one sens, czy nie ma w nich błędów takich jak naprzykład operator + dla niewłaściwych typów
  - moduł obsługi błędów, przy nieudanym sparsowaniu ciągu znaków podaje kolejne wykryte błędy użytkownikowi
  - tablica identyfikatorów razem z tymi zarezerwowanymi przez język i jej zarządzanie; używana przez moduły analizatora leksykalnego, składniowego oraz semantycznego

3 Gramatyka.

start ::= function
function ::= 'fun' type id '(' parameters? '):' body
parameters ::= type id (',' type id)*
body ::= (comment|assign |if | while |declare | return |functionCall |try)+
assign ::= id '=' expression
return ::= 'return' expression?
declare ::= type id ('=' expression)?
try ::= 'try:' body 'except' (exType 'as' id)?':' body
if ::= 'if' condition ':'body ('else:' body)?
while ::= 'while' condition ':' body
condition ::= relationalCondition (('and'|'or'), relationalCondition)*
relationalCondition ::= basicCondition, (('==' | '!='|'<' | '>'|'>='|'<='), basicCondition)*
basicCondition ::= '!'? ('(' condition')' |expression)
expression ::= term ([+-] expression)*
term ::= factor ([*/] term)*
factor ::= '"letter*'"|date|('-'? (number | '0'| id | '('expression')' | functionCall))
functionCall ::= id '(' arguments? ')'
arguments ::= expression (',' expression)*
id ::= letter (digit | letter)*
number ::= int | float
int ::= nonzeroDigit digit* | '0'
float ::= int '.' digit+
comment ::= '//'letter*
digit ::= nonzeroDigit | '0'
nonzeroDigit ::= [1-9]
letter ::= [A-Za-z/"]
type ::= 'int' | 'float'| 'date'|'string'
exType ::= 'ZeroDivisionError'
date ::= '(' [1-9] [0-9]{3}':' [0-1] [0-9] ':' [0-3] [0-9] ')'

4 Przykład kodu.

1)
```
fun int start():
    string napis = ”A”
    if napis ==”A”:
        print(”Tak”)
    else:
        int zmienna = 1
        while(zmienna <5):
            print(i)
            zmienna++
    return 0
```
2)
```
fun float divideByTwo(int x):
    try:
        return x/2
    except:
        return 0
fun int start():
    date a = (1999-05-30)
    date b = (2001-11-01)
    if(b –a >100):
        return divideByTwo(b-a)
    return 1
```
3)
```
fun int start():
    float a = 10.1
    float b = 0
    try:
        float c = a/b
    except ZeroDivisionError as e:
        print(e)
        return -1
    return c
```