Lipski Kamil Krzysztof
nr indeksu: 283689

Dokumentacja wstępna

Opis funkcjonalny

Język ma umożliwiać podstawowe przetwarzanie zmiennych zawierających wartości liczbowe z jednostkami SI z zakresu dwóch działów fizyki: kinematyki i dynamiki. Jednostki oraz ich relacje będą predefiniowane. Oprócz podstawowych zastosowań (tj. obliczania wartości wyrażeń fizycznych), język ma ułatwiać określanie poprawności wykonanych operacji (co zostało uzyskane właśnie przez wdrożenie do języka jednostek i ich relacji – obliczenia będą wykonywane nie tylko na wartościach zmiennych, ale też na jednostkach sprowadzanych do kombinacji jednostek podstawowych, tj. metrów, kilogramów i sekund). Język docelowo będzie językiem wsadowym.

Elementy wspierane przez język:

- instrukcja warunkowa typu `if - else`
- pętla typu `while`
- operatory: `+, -, *, /, (), !, &, |, ==, !=, >, >=, <, <=`
- definiowanie funkcji (bez możliwości rozdzielenia deklaracji i definicji)
- zmienne (w tym zmienne lokalne)
- typy: zmienna całkowitoliczbowa z jednostką lub bez, typ napisowy (nie w pełni wspierany, tylko do `print()`)
- stałe

Przykłady

1. deklaracja i definicja zmiennej:

```
var masa = 80 [kg];

var wspolczynnik = 2;
```

2. przykład konstrukcji `while` (obliczamy czas (w pełnych sekundach), po którym obiekt uzyska prędkość >= 100 m/s (przy zerowej prędkości początkowej)):

```
var v0 = 0 [m/s];
var v = 0 [m/s];
var a = 7 [m/s^2];
var t = 0 [s];

while (v < 100 [m/s])
{
    v = v + a * 1 [s];
    t = t + 1 [s];
}
```

3. przykład definicji funkcji + konstrukcja `if` w ciele funkcji (obliczamy drogę w ruchu jednostajnie zmiennym):

```
def liczDroge(var v0, var t, var a)
{
    var s = 0 [m];

    if (v0 < 0 [m/s] | t < 0 [s])
    {
        print("Nieprawidlowe wartosci argumentow funkcji");
        return 0 [m];
    }

    else
    {
        s = v0*t + a*t*t/2;
        return s;
    }
}
```

4. przykładowy program (z użyciem funkcji `liczDroge()`):

```
def liczDroge(var v0, var t, var a)
{
    …
}

var v0 = 100 [m/s];
var t = 60 [s];
var a = -1 [m/s^2];

var s = liczDroge(v0, t, a);

print("Wynik: ");
print(s);
```

Formalny opis gramatyki

```
KONWENCJE LEKSYKALNE

charString = '"', ..., '"' ;

id = (letter | '_'), {letter | digit | '_'} ;

digit = '0' | nonZeroDigit ;

nonZeroDigit = '1' | ... | '9' ;

number = '0' | nonZeroDigit, {digit} ;

unit = '[', ('m' | 'kg' | 's' | 'Hz' | 'N' | 'J' | 'W' | 'm/s' |
'm/s^2' | 'N*s' | 'J*s' | 'kg*m^2'), ']' ;
```

SKŁADNIA

```
Program = ProgStatement, {ProgStatement} ;

ProgStatement = FunDef | Statement ;

Statement = ConditionalStatement | SimpleStatement ;

ConditionalStatement = IfStatement | WhileStatement ;

SimpleStatement = (Assignment | FunCall | PrintCall |
ReturnStatement | VarDecl), ';' ;

FunDef = 'def', id, '(', [ParamList], ')', '{', Statement,
{Statement}, '}' ;

VarDecl = 'var', id, ['=', Expression] ;

IfStatement = IfBlock, [ElseBlock] ;

WhileStatement = 'while', '(', Expression, ')', '{', Statement,
{Statement}, '}' ;

Assignment = id, '=', Expression ;

FunCall = id, '(', [ArgList], ')' ;

PrintCall = 'print', '(', (charString | Expression), ')' ;

ParamList = Param, {',', Param} ;

Expression = OrExpr, {'|', OrExpr};

OrExpr = AndExpr, {'&', AndExpr} ;

AndExpr = RelExpr, {('==' | '!=' | '>' | '>=' | '<' | '<='},
RelExpr] ;

RelExpr = AdditiveExpr, {('+' | '-'), AdditiveExpr} ;

AdditiveExpr = MultiplicativeExpr, {('*' | '/'), MultiplicativeExpr}
;

MultiplicativeExpr = ['!' | '-'], OppExpr;

OppExpr = Constant | '(', Expression, ')' | FunCall | id ;

IfBlock = 'if', '(', Expression, ')', '{', Statement, {Statement},
'}' ;

ElseBlock = 'else', '{', Statement, {Statement}, '}' ;

ArgList = Arg, {',', Arg} ;

Param = 'var', id ;

ReturnStatement = 'return', Expression ;

Constant = number, [unit] ;

Arg = Expression;
```

Opis techniczny realizacji

Program zostanie napisany w języku C++. Będzie on uruchamiany za pośrednictwem terminala przez podanie nazwy i pliku wejściowego zawierającego kod do interpretacji, tj. poleceniem:

```
./nazwaProgramu PLIK_WEJSCIOWY [>PLIK_WYJSCIOWY]
```

Na wyjście (terminal bądź dookreślony przez użytkownika plik) będą przekazywane ciągi znaków wypisywane przez funkcję `print()` + to, co zostanie zwrócone za pomocą instrukcji `return` w głównym bloku programu.