Język Goose 1.0

Język Goose jest językiem imperatywnym podobnym do pythona.

Przykładowy:

Hello word:

```
def main () {
  printString("hello world");
  return 0;
}
```

Silnia na dwa sposoby:

```
def main () {
  printInt(fact(7)) ;
  printInt(factr(7)) ;
  return 0 ;
}

// iteracyjnie
def fact (n) {
  i = 1 ;
  r = 1 ;
  while (i < n+1) {
    r = r * i ;
    i++ ;
  }
  return r ;
}

// rekurencyjnie
def factr (n) {
  if (n < 2)
    return 1 ;
  else
    return (n * factr(n-1)) ;
}
```

Struktura programu

Program w języku Latte jest listą definicji funkcji. Na definicję funkcji składa się `def` nazwa, lista argumentów oraz ciało (nie zawiera typu zwracanej wartości). Funkcje mogą być przysłaniane przez inne fukncje o tej samej nazwie. W programie musi wystąpić funkcja o nazwie main zwracająca i nie przyjmująca argumentów

(od niej zaczyna się wykonanie programu). Funkcja może przyjmować argumenty z przedrostkiem `const`, wtedy zmienna jest tylko read-only.

```
 entrypoints Program ;
Program.   Program ::= [TopDef] ;
FnDef.     TopDef ::= "def" Ident "(" [Arg] ")" Block ;
separator nonempty TopDef "" ;
Arg.       Arg ::= Ident;
CntsArg.   Arg ::= "const" Ident;
separator  Arg "," ;
```

Instrukcje

Instrukcje: pusta,złożona,if,while,return jak w Pythonie, tylko na końcu wyrażenia musi być `;` .

```
 Block.     Block ::= "{" [Stmt] "}" ;
separator  Stmt "" ;
Empty.     Stmt ::= ";" ;
BStmt.     Stmt ::= Block ;
DeclCon.   Stmt ::= "const" Ident "=" Expr ;
DeclFun.   Stmt ::= "def" Ident "(" [Arg] ")" Block ;
Ass.       Stmt ::= Ident "=" Expr  ";" ;
TupleAss.  Stmt ::= Ident "=" "("[Expr] ")"  ";" ;
TupleAss1. Stmt ::= "(" [Arg] ")" "=" "("[Expr] ")"  ";" ;
TupleAss2. Stmt ::= "(" [Arg] ")" "=" Expr  ";" ;
Incr.      Stmt ::= Ident "++"  ";" ;
Decr.      Stmt ::= Ident "--"  ";" ;
Ret.       Stmt ::= "return" Expr ";" ;
RetTuple.  Stmt ::= "return" "(" [Expr] ")" ";" ;
VRet.      Stmt ::= "return" ";" ;
Cond.      Stmt ::= "if" "(" Expr ")" Stmt  ;
CondElse.  Stmt ::= "if" "(" Expr ")" Stmt "else" Stmt  ;
While.     Stmt ::= "while" "(" Expr ")" Stmt ;
For.       Stmt ::= "for" "(" Ident "=" Expr "to" Expr ")" Stmt ;
ForIn.     Stmt ::= "for" "(" Ident "in" Ident ")" Stmt ;
Break.     Stmt ::= "break" ;
Conti.     Stmt ::= "continue" ;
SExp.      Stmt ::= Expr  ";" ;
PrInt.     Stmt ::= "printInt" "(" Expr ")" ;
PrStr.     Stmt ::= "printStr" "(" Expr ")" ;
```

Typy

Nie można deklarować zmiennych bez przypisania od razu do niej wartości. Deklarowanie/inicializowanie zmiennych odbywa się bez deklaracji typu (tak jak w pythonie).

Przykład:

```
x = "napis";
y = 2;
```

Wyrażenia

Podzbiór zbioru wyrażeń dostępnych w Pythonie:

```
EVar.      Expr7 ::= Ident ;
ELitInt.   Expr7 ::= Integer ;
ELitTrue.  Expr7 ::= "true" ;
ELitFalse. Expr7 ::= "false"
EApp.      Expr7 ::= Ident "(" [Expr] ")" ;
EString.   Expr7 ::= String ;
EList.     Expr7 ::= "[" [Expr] "]" ;
EList1.    Expr7 ::= "[" Expr "]" "*" Expr ;
EAt.       Expr7 ::= Ident "[" Expr "]" ;
Neg.       Expr6 ::= "-" Expr7
Not.       Expr6 ::= "!" Expr7 ;
EMul.      Expr5 ::= Expr5 MulOp Expr6 ;
EAdd.      Expr4 ::= Expr4 AddOp Expr5 ;
ERel.      Expr3 ::= Expr3 RelOp Expr4 ;
EAnd.      Expr2 ::= Expr3 "&&" Expr2 ;
EOr.       Expr1 ::= Expr2 "||" Expr1 ;
ELambda.   Expr ::= "\\" [Arg] "->" Block;
ELambdaS.  Expr ::= "lambda" [Arg] "->" Block;
coercions  Expr 7 ;
separator  Expr "," ;
```

Wyrażenie logiczne zwracają typ `boolean` i są obliczane leniwie (drugi argument nie jest wyliczany gdy pierwszy determinuje wartość wyrażenia).

Predefiniowane funkcje

Są dostępne predefiniowane funkcje:

```
void printInt(int)
void printString(string)
void error()
void honk(int)
```

Funkcja `error` wypisuje `runtime error` i kończy wykonywanie programu. Funkcja `honk(int)` tworzy sygnał dźwiękowy.

Napisy

Napisy podobnie jak w Javie, czyli zmienne typu string zawierają referencję do napisu, zaalokowanego na stercie. Napisy moga występować jako: literały, wartości zmiennych, argumentów i wyników funkcji

Napisy mogą być użyte jako argumenty wbudowanej funkcji printString

Rozszerzenia

Tablice

Podobnie jak w Pythonie.

Przykłady:

```
 arr1 = [1]*3;
arr2 = [1,2,3,4,5,6,7,8];
const arr3 = ["A", true, 3];

\\ odwoływanie
arr1[2] == arr2[0];

\\ iterowanie
for (x in arr2) {
    printInt(x);
}
```

Deklaracje funckji

Podobnie jak pythonie.

Przykłady:

```
def foo(a, b, c) {
  def bar(x) {
    return a * x ;
  }
  return bar ;
}

  f = foo(2, 0, 0) ;
  g = lambda x -> {return x * 3; } ;
  h = \z -> {return z * 5 ;} ;
```

Break i contiune

Tak jak w pythonie.

Tuple

Podobnie jak pythonie.

Przykłady:

```
x = (1, 2);
(a, b, c) = (0, x[0], x[1]);
(e, d, f) = f(x);
```

Honk

Funkcja `honk(int)` służy do tworzenia krótkiego sygnału dźwiękowego. Może służyć do informowania użytkownika o skończeniu liczenia wartości. Argument `honk` to liczbma milisekund.

Przykład:

```
for (i = 1 to 1234567890) {
  foo(i);
}
honk(100);
```