JPP - Deklaracja języka do implementacji

Anna Jabłonowska

10 kwietnia 2022

1 Opis języka

Język imperatywny wzorowany na języku Latte.

Język posiada trzy typy (int, string, boolean). Język wymaga korzystania z funkcji zwracających wartość - nie posiada możliwości tworzenia procedur. Argumenty przekazywane do funkcji są przez wartość lub przez zmienną.

Użycie while oraz if, typy, arytmetyka i porównania, inicjowanie i przypisania zmiennych, tworzenie funkcji i ich wywoływanie, funkcje printInt() oraz printString() zapożyczone z języka Latte.

2 Tabelka funkcjonalności

Na 15 punktów
+ 01 (trzy typy)
+ 02 (literały, arytmetyka, porównania)
+ 03 (zmienne, przypisanie)
+ 04 (print)
+ 05 (while, if)
+ 06 (**funkcje** lub procedury, rekurencja)
+ 07 (**przez zmienną / przez wartość** / in/out)
08 (zmienne read-only i pętla for)

Na 20 punktów
+ 09 (przesłanianie i statyczne wiązanie)
+ 10 (obsługa błędów wykonania)
+ 11 (funkcje zwracające wartość)

Na 30 punktów
12 (4) (statyczne typowanie)
13 (2) (funkcje zagnieżdżone ze statycznym wiązaniem)
14 (1/2) (rekordy/listy/tablice/tablice wielowymiarowe)

15 (2) (krotki z przypisaniem)
16 (1) (break, continue)
17 (4) (funkcje wyższego rzędu, anonimowe, domknięcia)
18 (3) (generatory)

Razem: 20

3 Gramatyka mojego języka w notacji LBNF

Znacząca część gramatyki wzorowana na gramatyce języka Latte.

```
-- Programy (programs) ------------------------------------------------
entrypoints Program ;
Program.  Program ::= [TopDef] ;
FnDef.     TopDef ::= Type Ident "(" [Arg] ")" Block ;
separator nonempty TopDef "" ;
Arg.    Arg ::= Type Ident;
separator Arg "," ;

-- Instrukcje (statements) --------------------------------------------
Block.    Block ::= "{" [Stmt] "}" ;
separator Stmt "" ;
Empty.    Stmt ::= ";" ;
BStmt.    Stmt ::= Block ;
Decl.     Stmt ::= Type [Item] ";" ;
NoInit.   Item ::= Ident ;
Init.     Item ::= Ident "=" Expr ;
separator nonempty Item "," ;
Ass.      Stmt ::= Ident "=" Expr ";" ;
Incr.     Stmt ::= Ident "++" ";" ;
Decr.     Stmt ::= Ident "--" ";" ;
Ret.      Stmt ::= "return" Expr ";" ;
Cond.     Stmt ::= "if" "(" Expr ")" Stmt ;
CondElse. Stmt ::= "if" "(" Expr ")" Stmt "else" Stmt ;
While.    Stmt ::= "while" "(" Expr ")" Stmt ;
SExp.     Stmt ::= Expr ";" ;

-- Typy (types) -------------------------------------------------------
Int.      Type ::= "int" ;
Str.      Type ::= "string" ;
Bool.     Type ::= "boolean" ;
internal  Fun. Type ::= Type "(" [Type] ")" ;
separator Type "," ;

-- Wyrazenia (expressions) --------------------------------------------
EVar.     Expr6 ::= Ident ;
```

```
ELitInt.  Expr6 ::= Integer ;
ELitTrue. Expr6 ::= "true" ;
ELitFalse. Expr6 ::= "false" ;
EApp.     Expr6 ::= Ident "(" [Expr] ")" ;
EString.  Expr6 ::= String ;
Neg.      Expr5 ::= "-" Expr6 ;
Not.      Expr5 ::= "!" Expr6 ;
EMul.     Expr4 ::= Expr4 MulOp Expr5 ;
EAdd.     Expr3 ::= Expr3 AddOp Expr4 ;
ERel.     Expr2 ::= Expr2 RelOp Expr3 ;
EAnd.     Expr1 ::= Expr2 "&&" Expr1 ;
EOr.      Expr ::= Expr1 "||" Expr ;
coercions Expr 6 ;
separator Expr "," ;

-- Operatory (operators) ------------------------------------------------
Plus.     AddOp ::= "+" ;
Minus.    AddOp ::= "-" ;
Times.    MulOp ::= "*" ;
Div.      MulOp ::= "/" ;
Mod.      MulOp ::= "%" ;
LT.       RelOp ::= "<" ;
LE.       RelOp ::= "<=" ;
GT.       RelOp ::= ">" ;
GE.       RelOp ::= ">=" ;
EQ.       RelOp ::= "==" ;
NE.       RelOp ::= "!=" ;

-- Komentarze (comments) ------------------------------------------------
comment   "//" ;
comment   "/*" "*/" ;
```

4 Przykładowe programy

4.1 Funkcja Hello world

```
// Funkcja Hello world
int main(){
  string s = "Hello world";
  printString(s);
  return 0;
}
```

4.2 Funkcja wypisująca liczby parzyste do 10

```
// Funkcja wypisujaca liczby parzyste do 10
int main(){
  int i;
  i = 0;

  while(i <= 10){
    if(i % 2 == 0){
      printInt(i);
    }
    i++;
  }

  return 0;
}
```

4.3 Funkcja licząca k-tą liczbę Fibonacciego z wykorzystaniem rekurencji

```
// Funkcja liczaca k-ta liczbe Fibonacciego z wykorzystaniem rekurencji
int fibonacci(int k){
  if(k == 0 || k == 1){
    return k;
  } else{
    return (fibonacci(k - 1) + fibonacci(k - 2));
  }
}

int main(){
  int k = 5;

  printInt(k);
  printInt(fibonacci(k));

  return 0;
}
```