Struktura programu

Funkcje muszą zawsze coś zwrócić (nie ma typu void)

Funkcje muszą zwracać wartość za pomocą instrukcji return

Funkcje mogą być wzajemnie rekurencyjne

Program. Program ::= [TopDef] ;

FnDef. TopDef ::= Type Ident "(" [Arg] ")" Block ;

separator nonempty TopDef "" ;

Arg. Arg ::= Type Ident;

separator Arg "," ;

Instrukcje

Instrukcje: pusta, złożona, if, while, przypisanie

Instrukcja for jak w pascal’u (na in’tach)

Brak instrukcjami, postinkrementacja, postdekrementacja

każda zmienna musi być zadeklarowana przed użyciem.

inicjalizowana wartością domyślną (0 dla int, "" dla string, false dla bool).

Zmienne zadeklarowane w bloku nie są widoczne poza nim i przesłaniają zmienne o tej samej nazwie spoza bloku. W obrębie bloku zmienne muszą mieć unikalne nazwy.

Nie może być zmiennej oraz funkcji o tej samej nazwie

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 ";" ;

Ret. Stmt ::= "return" Expr ";" ;

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 ;

SExp. Stmt ::= Expr ";" ;

Typy

Nie ma konwersji pomiędzy typami.

Int. Type ::= "int" [ " const"] ;

Str. Type ::= "string"[ " const"];

Bool. Type ::= "boolean"[ " const"];

internal Fun. Type ::= Type "(" [Type] ")" ;

separator Type "," ;

Wyrażenia

Wyrażenia są obliczane zachłannie

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 ;

Napisy

Napisy czyli zmienne typu string zawierają referencję do napisu, zaalokowanego na stercie.

Napisy mogą występować jako: literały, wartości zmiennych, argumentów

Napisy mogą być użyte jako argumenty wbudowanej funkcji printString

Napisy mogą być konkatenowane przy pomocy operatora +. Wynikiem tej operacji jest nowy napis będący konkatenacją argumentów

Predefiniowane funkcje

void printInt(int)

void printString(string)

Parametry funkcji

Wszystkie parametry są przekazywane przez wartość. Wewnątrz funkcji parametry formalne zachowują się jak zmienne lokalne (czyli przypisania na nie są dozwolone).

Przekazanie napisu jako parametru odbywa się poprzez przekazanie przez wartość referencji do napisu.

Na 15 punktów  
1. Co najmniej trzy typy wartości: int, bool i string  
(to znaczy if 2+2 then \_ parsuje się, ale wyrażenie ma niepoprawny typ).  
2. Literały, arytmetyka, porównania.  
3. Zmienne, operacja przypisania  
4. Jawne wypisywanie wartości na wyjście (instrukcja lub wbudowana procedura print).  
5. while, if z else   
6. Funkcje lub procedury (bez zagnieżdżania), rekurencja.  
  
~~7. co najmniej dwa sposoby przekazywania parametrów (przez zmienną / przez wartość / in/out),~~  
8. zmienne „read-only” i użycie ich np. w implementacji pętli for w stylu Pascala  
(for i = pocz to kon - wewnątrz pętli nie można zmienić wartości zmiennej sterującej,wartość kon liczona tylko raz - przed wejściem do pętli)  
Na 20 punktów  
9. Przesłanianie identyfikatorów ze statycznym ich wiązaniem (zmienne lokalne i globalne ~~lub  
zagnieżdżone procedury/funkcje~~).  
10. Obsługa błędów wykonania, np. dzielenie przez zero (może być elegancki komunikat i  
zatrzymanie interpretera).  
11. Funkcje przyjmujące i zwracające wartość dowolnych obsługiwanych typów (tzn. nie tylko procedury; za to mogą być tylko funkcje – jak w języku C).  
Do 30 punktów wg cennika...  
12. Statyczne typowanie (tj. zawsze terminująca faza kontroli typów przed rozpoczęciem  
wykonania programu) – 4pkt,  
13~~. Dowolnie zagnieżdżone definicje funkcji / procedur z zachowaniem poprawności statycznego  
wiązania identyfikatorów (jak w Pascalu) – 2 pkt,~~  
~~14. Rekordy albo tablice indeksowane int albo coś à la listy – 1pkt,  
ALBO tablice wielowymiarowe przekazywane i przypisywane "przez wskaźnik" (jak w Javie), a  
nie "przez kopię" – 2pkt~~  
~~15. Dowolnie zagnieżdżone krotki z przypisaniem jak w Pythonie (składnia wedle uznania) – 2pkt,~~  
~~16. Operacje przerywające pętlę while - break i continue – 1pkt,~~  
~~17. Funkcje jako parametry,  
zwracanie funkcji w wyniku, domknięcia à la JavaScript.  
funkcje anonimowe – 4pkt,~~  
~~18. Procedury generujące i składnia do ich używania (np. jak w Pythonie - instrukcja yield oraz  
next, a także for x in generator(...) ) – 3pk~~