Deklaracja języka

Wiktoria Walczak, nr indeksu: 429656

a) gramatyka języka

Gramatyka znajduję się w pliku gramar.cf

b) przykładowe programy

Poprawne programy znajdują się w folderze good, niepoprawne programy znajdują się w folderze bad.

c) opis języka

Główne cechy:

- język imperatywny,
- wzorowany na języku Latte,
- ze statycznym typowaniem.
- struktura:
    1. deklaracje zmiennyc globalnych i funkcji
        - funkcje mogą odwoływać się do funkcji globalnych zadeklarowanych w dowolnym miejscy programu
        - funkcje mogą się odwoływać tylko do zmiennych globalnych, które zostały zadeklarowane powyżej ich deklaracji
    2. main
        - jest funkcją typu void
        - nie przyjmuje argumentów
        - jest zdefiniowany na samym końcu programu

```
def void main () {
  # ...
  return;
}
```

Zmienne

- cztery typy wartości: int, bool, string, func

- func to typ funkcyjny posiadającu dwa parametry - typ zwracanej wartości i typy argumentów funkcji

```
var func [<typ zwracanej wartości>] [<typy argumentów>] f;
```

- słowo kluczowe “var” do deklarowania zmiennych:

```
var <typ zmiennej> <nazwa zmiennej> = <wartość zmiennej>;

var int a = 4;
var bool b = true;
var string c = "ala";
```

- są zmienne lokalne i globalne (Przesłanianie identyfikatorów ze statycznym ich wiązaniem → /good/09-zmienneGlob)
- zmienne globalne mogą być zadeklarowane w dowolnym miejscu (powyżej funkcji main), jednak są widoczne tylko poniżej ich deklaracji

Funkcje

- słowo kluczowe “def” do definiowania funkcji (jak w Pythonie):

```
def <typ zwracanej wartości> <nazwa funkcji> () {}
```

- aby wywołać funkcję należy użyć słowa kluczowego “fn”:

```
fn <nazwa funkcji> (<argumenty>);
```

- funkcje przyjmują i zwracają wartość dowolnych obsługiwanych typów
- dwa sposoby przekazywania parametrów: przez zmienną i przez wartość (good/07-zmienna_wartosc)

```
def <typ zwracanej wartości> <nazwa funkcji> (<typ argumentu> <nazwa argumentu>) { ... }
```

```
def <typ zwracanej wartości> <nazwa funkcji> (ref <typ argumentu> <nazwa argumentu>) { ... }
```

- funkcje mogą być wzajmenie rekurencyjne, widzą definicje wszystkich funkcji, i tych zdefiniowanych powyżej i poniżej (dotyczy funkcji globalnych)

- dowolne zagnieżdżanie definicji funkcji (good/12-zagniezdzone_funkcje)
- każda funkcja musi zawierać return:
  - typ zwracanej wartości musi zgadzać się z deklaracją funkcji
  - return musi znajdować się na końcu bloku
  - jeden return na jedną funkcję
  - return nie może zjadować się w zagnieżdzonych blokach (while i if)
- dostępna jest predefiniowana funkcja print

```
fn print("hello");
```

- można tworzyć funkcje anonimowe

```
def <typ zwracanej wartości> (<typy argumentów>) { ... };
```

  - żeby wywołać funkcję anonimową, trzeba ją najpierw przypisać na zmienną

```
var func[void][] wypisz = def void() { fn print("a"); return; };
fn wypisz();
```

- wszystkie funkcje mogą być przekazywane jako parametry do funkcji oraz mogą być zwracane w wyniku
- tylko funkcje anonimowe i lokalne mogą być przekazywane do funkcji przez referencję, funkcji globalnch nie można przekazywać jako referencji
- definicji funkcji globalnych nie wolno zmieniać

Pętle

- break i continue (good/16-break_continue)

d) tabela cech

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
Do 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	31