TKOM – Projekt Wstępny

Język do operacji na walutach

Czobot Stanisław

295797

1. Opis funkcjonalny

Istotą projektu jest stworzenie język programowania, który umożliwi operacje na zmiennych “walutowych”. Każda z dostępnych walut będzie reprezentowana jako oddzielny typ danych, nazwami poszczególnych typów danych będą międzynarodowe skróty walut jakie będą one reprezentować.

2. Walutowe typy danych

Na potrzeby projektu będzie zrealizowanych 6 walutowych typów danych, tj:

a. PLN – Złoty
b. EUR – Euro
c. USD – Dolar amerykański
d. GBP – Funt szterling
e. CHF – Frank szwajcarski
f. RUB – Rubel rosyjski

Przykładowa deklaracja zmiennej typu walutowego będzie wyglądać następująco:

```
EUR nazwa_zmiennej = 10;
```

Tak zadeklarowana zmienna będzie reprezentowała walutę Euro w ilości 10€.

Konwersje pomiędzy różnymi walutami będą odbywać się na podstawie przeliczników, które umieszczone zostaną w oddzielnym pliku (w pliku tym dodatkowo możliwe będzie zdefiniowanie własnej waluty), zapisane w postaci tabeli (przeliczników tych nie będzie można dynamicznie zmieniać podczas działania programu). Dodatkowymi typami danych będą typ bool(true lub false) oraz zmienne liczbowe: całkowitoliczbowe (int) i zmiennoprzecinkowe(double).

3. Funkcjonalności języka

Funkcjonalności związane z walutowymi typami danych:

- Dodawanie, odejmowanie różnych typów walutowych

  Np.:

  ```
  EUR eur = 10;

  PLN pln = 2;

  USD usd = eur + pln;
  ```

  Tak zapisany kod przypisze zmiennej *usd* wartość odpowiadającą sumie zmiennych *eur* i *pln* przemnożonych przez odpowiednie przeliczniki dotyczące tych walut.

- Mnożenie, dzielenie typów walutowych przez zmienne liczbowe

  Np.:

  ```
  int x = 3;

  GBP jedenFunt = 1;

  GBP trzyFunty = jedenFunt * 3;
  ```

  Ze względu na jednostki walutowe nie możliwe będą operacje mnożenia dwóch zmiennych walutowych oraz dzielenia zmiennej liczbowej przez zmienną walutową, tj.:

  ```
  GBP niedozwolone_mnozenie = jedenFunt * trzyFunty;

  int niedozwolone_dzielenie = x / trzyFunty;
  ```

- Uzyskanie wartości liczbowej zapisanej wewnątrz zmiennej walutowej(operacja taka umożliwi wykonać operacje takie jak te zapisane powyżej dodając wywołanie ogólnodostępnej funkcji value).

  Np.:

  ```
  GBP dozwolone_mnozenie = jedenFunt * amount(trzyFunty);
  ```

  W tak zapisanym kodzie, z wykorzystaniem przypadku z poprzedniego podpunktu, funkcja amount(trzyFunty) zwróci wartość liczbową równą 3, przez co umożliwi pośrednie mnożenie dwóch zmiennych walutowych.

- Porównania dwóch zmiennych walutowych, zwracających wartość logiczną prawda lub fałsz w zależności od tego czy spełniony jest warunek po sprowadzeniu do wspólnej waluty.

Np.:

```
EUR eur = 1;
PLN pln = 4;
if( eur > pln )
{
      ...
}
```

Nie możliwe będzie wykonanie operacji porównania pomiędzy zmienną liczbową i zmienną walutową.

- Konwersja między walutami - dostępna będzie domyślna podczas wykonywania operacji arytmetycznych i logicznych na zmiennych walutowych jak w przykładach powyżej.

Ogólne funkcjonalności języka:

- Instrukcje warunkowe (konstrukcja *if*)
- Instrukcje pętli (konstrukcja *while*)
- Deklarowanie zmiennych walutowych oraz liczbowych
- Definiowanie własnych funkcji – funkcje będą mogły przyjmować parametry wejściowe oraz zwracać wartość zadanego typu, np.:

```
EUR suma( EUR eur, USD usd)
{
      return eur + usd;
}
```

- Drukowanie łańcuchów znaków, wartości liczbowych, zmiennych walutowych w konsoli, za pomocą funkcji print().

4. Formalny opis gramatyki

program = { function };

function = signature parameters code_block;

signature = data_type id;

parameters = ‘(‘ [signature { ‘,‘ signature }*] ‘)’;

code_block = ‘{‘ line [line]* ‘}’;

line = (statement ’;’) | (if_statement | while_statement);

statement = function_call | declaration | expression;

declaration = data_type id [‘=’ expression][‘,‘ id [‘=’ expression]]*;

expression = id | arithmetic_operation | value | function_call;

arithmetic_operation = value_instance {aritmetic_operator value_instance}*;

arithmetic_operator = ‘+’ | ‘-’ | ‘*’ | ‘/’;

value_instance = id | value | funcion_call;

if_statement = ‘if’ ‘(’ condition ‘)’ line_or_block;

while_statement = ‘while’ ‘(’ condition ‘)’ line_or_block;

line_or_block = line ’;’ | code_block;

condition = condition_operation [condition_linker condition_operation]*;

condition_operation = (expression condition_operator expression) | bool_value | expression;

condition_operator = ‘==’ | ‘!=’ | ‘<’ | ‘>’ | ‘<=’ | ‘>=’;

condition_linker = ‘&&’ | ‘||’;

function_call = id ‘(’ call_elements ‘)’;

call_elements = expression [‘,’ expression]*;

data_type = ‘int” | ‘double’ | ‘bool’ | ‘PLN’ | ‘USD’ | ‘EUR’ | ‘GBP’ | ‘CHF’ | ‘RUB’;

bool_value = ‘true’ | ‘false’;

value = number | ‘true’ | ‘false’;

number = not_zero_digit [digit];

not_zero_digit = ‘1’ | ‘2’ | ‘3’ | ‘4’ | 5’ | 6’ | ‘7’ | ‘8’ | ‘9’;

digit = ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | 5’ | 6’ | ‘7’ | ‘8’ | ‘9’;

id = letter { digit | letter };

letter = ‘a’ | ‘b’ | ‘c’ | ‘d’ | ‘e’ | ‘f’ | ‘g’ | ‘h’ | ‘i’ | ‘j’ | ‘k’ | ‘l’ | ‘m’ | ‘n’ | ‘o’ | ‘p’ | ‘r’ | ‘s’ | ‘t’ | ‘u’ | ‘v’ | ‘w’ | ‘x’ | ‘y’ | ‘z’ | ‘A’ | ‘B’ | ‘C’ | ‘D’ | ‘E’ | ‘F’ | ‘G’ | ‘H’ | ‘I’ | ‘J’ | ‘K’ | ‘L’ | ‘M’ | ‘N’ | ‘O’ | ‘P’ | ‘R’ | ‘S’ | ‘T’ | ‘U’ | ‘V’ | ‘w’ | ‘X’ | ‘Y’ | ‘Z’ |;

5. Opis techniczny realizacji

Struktura pliku z walutami:

	EUR	PLN	USD	GBP	CHF	RUB
EUR	1	4	1,25	...		
PLN	0,25	1	0.27			
USD	0,8	3.8	1			
GBP	...			1		
CHF					1	
RUB						1

Skróty będące nazwami typów danych mogą składać się jedynie ze znaków ASCII od ‘A’ do ‘Z’.

Projekt zostanie zrealizowany w języku Java.

Program będzie uruchamiany z poziomu terminala, do jego uruchomienia potrzebne będzie podanie dokładnie dwóch argumentów wejściowych, tj.: nazwa pliku zawierającego kod programu napisanego w tworzonym języku, nazwa pliku zawierającego tabelę z przelicznikami pomiędzy walutami.