### Dokumentacja wstępna TKOM – „Język do operacji na listach”

### Ewelina Chmielewska

### 283714

1. **Treść zadania**

„Język do operacji na listach”

Język zorientowany wokół struktur listowych i operacjach na nich, takich jak filtrowanie, złączenia, dostęp do podlist etc.

1. **Opis ogólny**

Celem projektu jest utworzenie języka umożliwiającego operowanie na listach, który swoją składnią będzie zbliżony do języka C++. Język ten powinien umożliwiać przede wszystkim filtrowanie zawartości list, złączenia ich oraz dostęp do poszczególnych ich elementów. Efektem prac ma być utworzenie programu, który otrzymawszy plik z poprawnie zapisanym ciągiem poleceń w tworzonym języku, właściwie zinterpretuje i wykona zadane komendy. W przypadku wystąpienia błędu leksykalnego lub składniowego program powinien wyświetlać komunikat o błędzie, wskazujący pozycję, na której błąd wystąpił.

1. **Typy i operacje**

W projekcie będą dostępne 3 główne typy danych:

* **list** – struktura listowa, umożliwiająca tworzenie zbioru elementów typu number. Dopuszczalne jest utworzenie listy pustej. Na obiekcie listy dokonać można różnych operacji, których sposób wywołania jest następujący:

*lista.operacja(opcje)*

*lista* – identyfikator listy (nazwa wcześniej zdefiniowanej listy) lub lista w postaci standardowej (np. [1, 2, 3, 4])

*operacja* – jedna z możliwych do wykonania operacji, których listę zamieszczono niżej

*opcje* – opcje zależne od wykonywanej operacji. Konieczne do zawarcia opcje zamieszczono niżej.

Operacje możliwe do wykonania na obiektach typu list to:

1. **filter(*warunek*)** – zwraca nową listę, składającą się z elementów listy przefiltrowanej, które spełniły zadany *„warunek”*. Dokładna postać warunku opisana jest w zamieszczonej w sekcji „Gramatyka” gramatyce.
2. **each(*operacja*)** – wykonuje *„operację”*  na wszystkich elementach listy. Dokładna postać operacji opisana jest w zamieszczonej sekcji „Gramatyka” gramatyce.
3. **get(*indeks*)** – zwraca element listy o zadanym *„indeksie”* (indeks musi być numerem)
4. **length()** – nie przyjmuje żadnych argumentów. Zwraca długość listy
5. **delete(*indeks*)** – usuwa element listy o zadanym *„indeksie”* (indeks musi być numerem)

* **number** – typ danych dotyczący liczb całkowitych. Analogiczny do typu int w języku C++.
* **bool** – logiczny typ danych, składający się z dwóch elementów: prawda, fałsz. Analogiczny do typu bool w języku C++.

1. Testowanie
2. Gramatyka:

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

Opis funkcjonalności:

1. Tworzenie i modyfikacja list

* łączenie list
* dodawanie elementów do list
* usuwanie elementów z listy
* modyfikacja elementów listy

1. Przeglądanie list

* wyświetlanie zawartości list

1. Wyszukiwanie

* zwracanie elementów listy spełniających określony warunek

Dodatkowe założenia:

* Typowanie: silne
* Wyrażenia z priorytetem operatorów (1. \*, / 2. +, -, 3. =)
* Kolejność wykonywania operacji na liście: od lewej do prawej

np. a.filter(…).each(…) – najpierw filtrowanie, potem each

Opis techniczny:

Język programowania: Python

Uruchomienie aplikacji: z terminala

Przykłady w języku:

1. Deklaracja

* number x = 1;
* list y = [1, 2 , 3];
* bool z = true;
* number z = [1, 2, 3, 4].filter(x > 2) 🡺 ciężko jest przewidzieć jaki będzie ostateczny wynik wykonywanej operacji, więc mimo oczywistego błędu taki zapis jest dopuszczalny. Zaproponowana gramatyka nie obsługuje tego typu sytuacji, więc trzeba będzie obsłużyć to w inny sposób

1. Funkcja zwracająca prostą listę

list foo(){ return [1, 2, 3, 4]}

1. Funkcja zwiększająca wszystkie wartości w tablicy o 1. Zwraca listę

list foo(list a){

a = a.each(+1);

return a;

}

1. Zastosowanie filtra

list a = [1, 2, 3, 4, 5, 6];

a = a.filter(x > 2 & x < 5);

print(a);

Zwrócony wynik: [3, 4]

list a = [1, 2, 3, 4, 5, 6];

a.filter(x > 2 & x < 5);

print(a);

Zwrócony wynik: = [1, 2, 3, 4, 5, 6]

1. Prosty program wykorzystujący funkcję zliczającą elementy mniejsze lub równe maksymalnej wartości w dwóch listach

number foo(list a, list b, number maxVal){

list c = a.filter(x < maxVal + 1) + b.filter(x < maxVal + 1);

return = c.length();

}

main(){

list a = [1, 2, 3, 4, 5];

list b = [1, 2, 3, 4, 5];

number maxVal = 4;

number result = foo(a, b, maxVal);

print(result);

return 0;

}