Projekt TKOM Mariusz Pakulski

Podstawowe cechy:

- Typowanie silne, dynamiczne
- Interpreter napisany w języku C++

Czy można zmieniać ustalenia projektowe takie jak typowanie silne albo język programowania C++?

Temat:

Język z wbudowanym typem słownika z określoną kolejnością elementów.
Kolejność elementów w słowniku jest określana w momencie jego tworzenia – argumentem konstruktora słownika jest funkcja określająca, w jakiej kolejności mają być wstawiane elementy.
Możliwe są podstawowe operacje na słowniku (dodawanie, usuwanie, wyszukiwanie elementów wg klucza, sprawdzanie, czy dany klucz znajduje się w słowniku itd.), iterowanie po elementach oraz wykonywanie na słowniku zapytań w stylu LINQ.

Krótki opis działania:

Słownik będzie można stworzyć albo pusty albo od razu podając kilka jego wartości. (Zastanawiam się na ile to utrudni, że można podac od razu kilka wartości). Wtedy trzeba będzie podać funkcję porównującą po kluczach.

W przypadku gdy później chcelibyśmy dodać nową wartość do słownika za pomocą insert to będzie ona korzystała z tej funkcji porównującej po kluczach.

Możliwości programisty:

1. obsługuje podstawowe typy danych liczbowych

będzie możliwe użycie typów całkowitoliczbowych i zmiennoprzecinkowych, wyrazów, stałych znakowych.

2. operacje matematyczne o różnym priorytecie wykonania, obsługa nawiasów

Programista będzie mógł dodawać, odejmować, mnożyć, dzielić, obliczać resztę z dzielenia liczby dowolnego tego samego typu. Priorytetami operatorów będą w kolejności od najwyższego:

- Reszta z dzielenie
- Mnożenie i dzielenie
- Dodawanie i odejmowanie

W pierwszej kolejności będzie wykonywane to co w nawiasach okrągłych (tylko takie nawiasy będą wykorzystywane).

Na stringu będzie możliwe wykonanie konkatencji.

3. Operacje logiczne pomiędzy zmiennymi tego samego typu takie jak <, >, ==, !=, <=, >=

4. String może zawierać dowolne znaki, włącznie z wyróżnikiem stringa (np. cudzysłów lub apostrof), nową linią, znakiem tabulacji
5. obsługuje komentarze, będą one się zaczynały :], a kończyły się [: kiedy będą wielolinijkowe , kiedy jednolinijkowe to będą //.
6. pozwala tworzyć zmienne, przypisywać do nich wartości i odczytywać je

Tworzenie zmiennych: typ_zmiennej nazwa_zmiennnej

Przypisywanie wartości: nazwa_zmiennej = wartość

Odczytywanie wartości: nazwa_zmiennej

- typowanie dynamiczne
- typowanie silne
- mutowalne

7. zakresy widoczności zmiennych

Nie ma zmiennych globalnych, zmienne są widziane tylko w tym samym węźle drzewa wywołań.

Zmienne mogą być tylko publiczne, nie zakładam tworzenia klas.

8. Argumenty będą przekazywane do funkcji przez wartość
9. Programista będzie mógł tworzyć instrukcje warunkowe, pętle, funkcje.
10. Rekursywne wywołania funkcji

Sposób tworzenia i zapamiętywania funkcji w interpreterze jest przedstawiony w dalszej części.

11. obsługa błędów

Zostaną obsłużone różne wyjątki takie jak błąd dzielenia przez 0, błędy leksykalne (np. ciąg znaków nie został dopasowany do żadnego tokenu), składniowe (wyrażenie niezgodne z regułami gramatyki, niezgodne typy), semantyczne (nie istnieje zmienna użyta w wyrażeniu)

Funkcje, które będą wywoływane na obiekcie słownika:

1. Dodawanie do słownika
2. Usuwanie ze słownika
3. wyszukiwanie elementów wg klucza,
4. sprawdzanie, czy dany klucz znajduje się w słowniku,
5. sprawdzenie liczby elementów w słowniku
6. lower bound - znajduje pierwszy niemniejszy element niż podany,
7. upper bound - znajduje pierwszy większy element niż podany,
8. iterowanie po kolejnych elementach – funkcja, która by zwracała iterator do następnego elementu podając w argumencie iterator (wskaźnik) poprzedniego elementu.
   W przypadku gdy już nie będzie kolejnego elementu funkcja ta zwracałaby specjalny iterator końca. (wskazujący na pozycję za ostatnim elementem)
9. iterowanie po poprzednich elementach – na odwrót
   Iterator będzie wskazywał na pozycję przed pierwszym elementem.
10. Zmiana wartości o podanym kluczu – opcjonalne, bo można usunąć i dodać

Funkcje stworzone przez programistę:

1. Funkcja ustanawiająca który element jest mniejszy, będzie przyjmowała dwa argumenty, które będą typów, jakie będą miały klucze w słowniku. Kolejność dla intów może być oceniona na podstawie wartości, a dla stringów na podstawie pierwszej litery.
2. Opcjonalna funkcja porównywania wartości po wartościach (może użyć tej samej co po kluczach jeśli wartości są tego samego typu co klucz lub w ogóle nie chcieć sortować po wartościach). Będzie przyjmowała dwa argumenty, które będą typów jakie maja wartości w słowniku.

Zapytania o słownik:

Funkcje select razem z opcjonalnym where, obowiazkowym from, opcjonalnym order by (razem z opcjonalnym desc (malejąco), bo domyślnie będzie rosnąco), opcjonalnym przycięciem pierwszych wyników.

Order by by sortował albo po kluczach albo po wartościach. Miałby podawaną funkcję sortującą jako argument. Jest to jedna z opcjonalnych funkcji stworzonych przez programistę. Domyślnie sortuje po kluczach. Gdyby sortował po wartościach to należy napisać Order by values.

Czy programista miałby używać group by?

Po klauzuli select może pojawić się Sum, Average.

W jaki sposób funkcja będzie zapisywana przez kompilator:

Jak zapamiętać funkcję zdefiniowaną w wymyślonym języku do programu w C++ w trakcie jego uruchomienia?

Czy można stworzyć w trakcie uruchomienia programu funkcję w C++ z wymyślonego języka?

Mój wymyślony sposób poniżej. Dzięki temu jest spełniony warunek rekursywnego wywołania funkcji.

Każda funkcja będzie klasą.

Całą funkcję będziemy przechowywali w zbiorze (set), zakładam, że funkcje będą takie same jak będą miały taką samą nazwę, argumenty i zwracany typ. Nie można utworzyć dwóch takich samych funkcji w programie.

Zmienne przekazywane do konstruktora:

- Typ zwracany funkcji – identyfikator typu (z poziomu C++ to będzie enum)
- Nazwa funkcji – string
- Lista argumentów funkcji – wektor dwuelementowych krotek
- Kod w postaci tekstowej - string

Metody:

- Eval – służy to wywołania funkcji, Najpierw sprawdza czy tą funkcję można wywołać z podanymi argumentami i zwraca wynik.
  W metodzie eval będzie tworzony lokalny interpreter, który będzie interpretował zmienną funkcji„kod funkcji w postaci tekstowej”, który będzie typu string. Dzięki temu będziemy

mogli zapisać funkcję jako daną i wykonywać ją dopiero w momencie gdy tego potrzebujemy.

- Operator porównujący (mniejszości) – Sprawdza czy już dwie funkcje są „równe” (bierzemy pod uwagę typ zwracany funkcji, nazwę funkcji i listę argumentów funkcji)

Pytanie:

Czy typowanie statyczne od silnego różni się tym, że w typowaniu statycznym nie można zmienić typu danej zmiennej (która by miała pozostać o takim typie na stałe), a w typowaniu silnym nie można podczas wykonywania operacji arytmetycznych zmienić typu danej zmiennej chwilowo na czas wykonywania operacji tylko po to, aby mogła być wykonana ta operacja, gdy później typ by pozostał taki jak na początku?

Czy należy zrobić sumowanie słowników albo inne operacje pomiędzy nimi?