



JPS Ćw 3

Stos rezultatów

QRES W ZAŁOŻENIACH SBA

Stos rezultatów w SBA jest używany do przechowywania tymczasowych wyników podzapytań przy obliczaniu wyniku ostatecznego zapytania. Działa jak klasyczny stos i zasada jego działania jest wzięta bezpośrednio z tradycyjnych języków programowania.

RODZAJE REZULTATÓW W SBA

W SBA zakładamy, że zapytanie nigdy nie zwraca obiektu, lecz referencje do obiektu (w rzeczywistości wystarczy, że zwróci OID). Obiekty składowane są w DataStore (patrz C2) i każda referencja zwrócona przez zapytanie powinna mieć odzwierciedlenie w DataStore i na jej podstawie jesteśmy w stanie uzyskać „wnętrze” obiektu.

Tak samo jak w innych językach programowania, SBA ma możliwość zwrócenia struktury, worka, sekwencji oraz wartości atomowej. Dodatkowo, SBA pozwala na zwrócenie struktury złożonej z nazwy i wartości, zwanej „binder-em”.

Możliwe opcje:

- **wartość atomowa**

np. 5, 3.5, „Ala ma kota”, true

- **referencja**

np. i0

- **struktura**

np. struct(„Marcin”, „Dąbrowski”, 1, 2, 3, 4)

- **bag**

np. bag(1, 2, i0, „tekst”)

- **sekwencja**

np. sequence(1, 2, 3, 4, 5)

- **binder**

np. imię(i0), x(bag(1, 2, 3))



Założenia:

Jeśli **x1**, **x2**, **x3** są możliwymi rezultatami zapytania, to również **bag(x1, x2, x3)** jest poprawnym rezultatem zapytania. To samo dotyczy **struktury**, **sekwencji** oraz **bindera**.

Na potrzeby naszego projektu, nie ma potrzeby rozróżnienia między bag a sequence. Wystarczy typ **bag** i pamiętanie, że elementy w wewnętrznej liście kolekcji są poukładane w pewnej kolejności, która, być może, ma dla naszego zapytania znaczenie.

FUNKcjONALNOŚĆ QRES

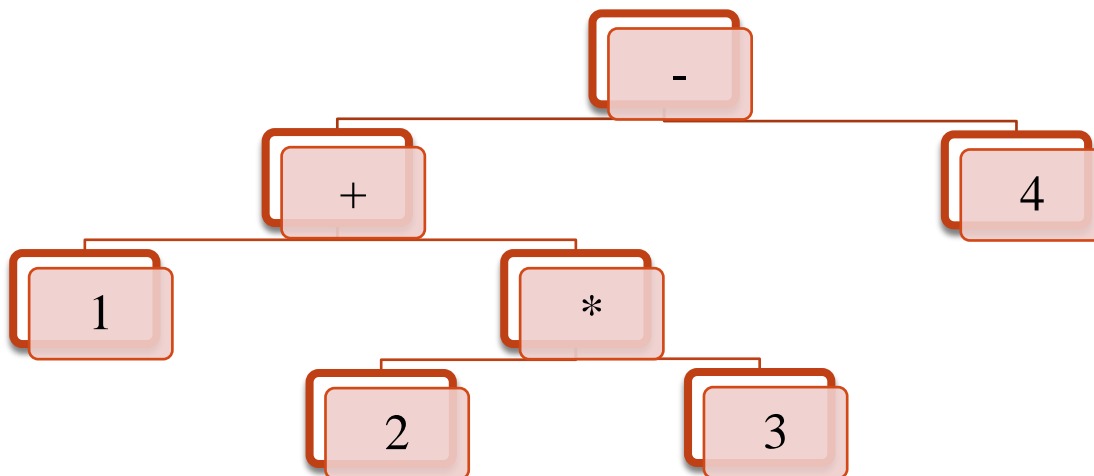
- Służy do tymczasowego przechowywania rezultatów zapytań
- Składa się z elementów będących rezultatami zapytań.
- Każdy element na stosie jest pojedynczym rezultatem (choć może to być element który jest kolekcją).
- Operacja push umieszcza nowy element na szczycie stosu
- Operacja pop usuwa element ze szczytu stosu

PRZYKŁAD – ZAPYTANIE BEZ NAZWY

Zapytanie typu:

$$1+2*3-4$$

Drzewo składniowe tego zapytania wygląda następująco:

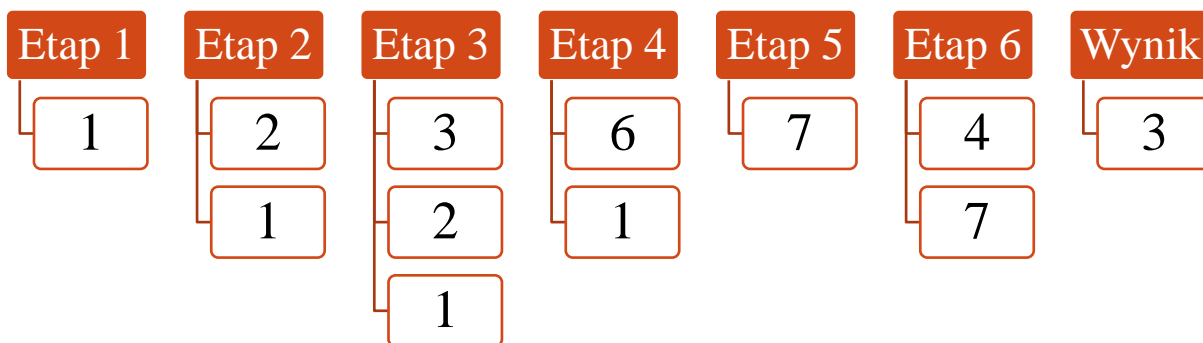


Co to jest drzewo składniowe i jak się je buduje dowiemy się później. Teraz wystarczy, że poznamy prosty mechanizm chodzenia po takim drzewie:

```

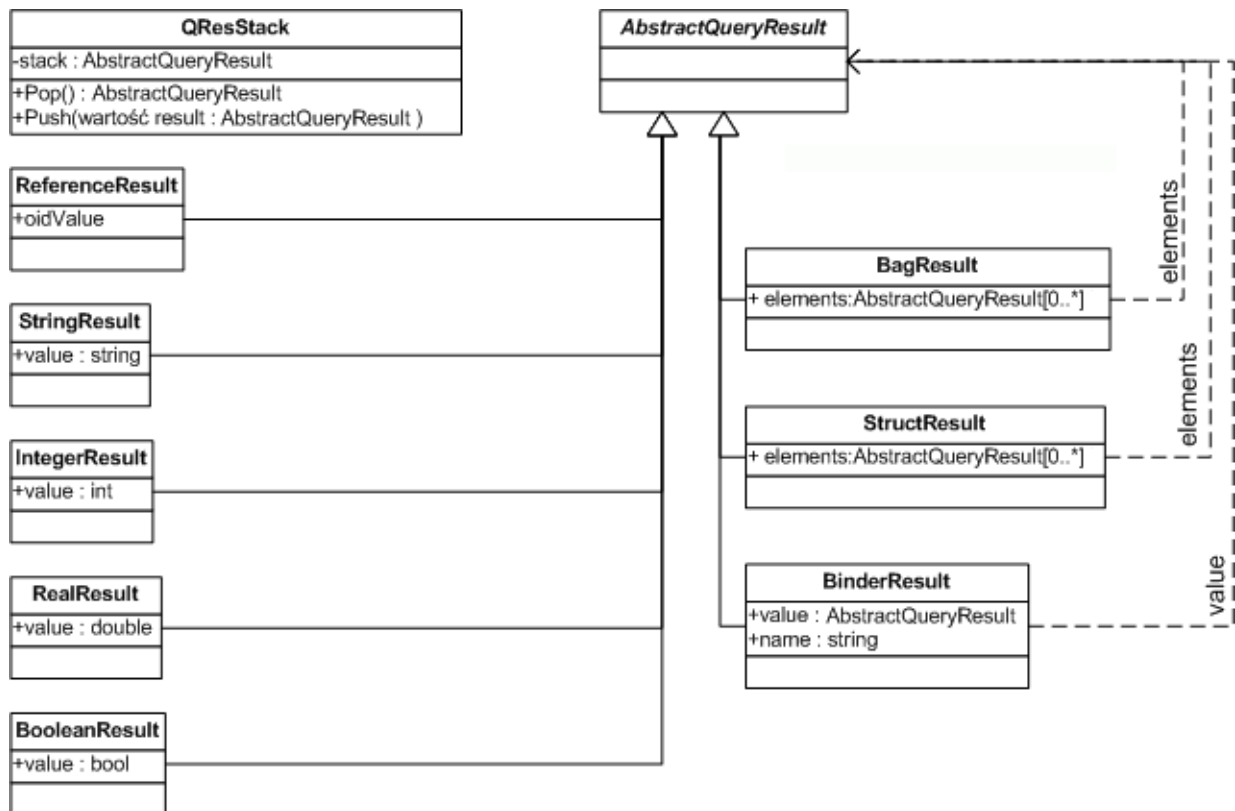
tree_walk(node) {
  if (node is leaf)   QRES.push(leafValue);
  else if (node is operator) {
    tree_walk(node.left);
    tree_walk(node.right);
    Result rightR = QRES.pop();
    Result leftR = QRES.pop();
    // wykonaj operację związaną z operatorem np.
    Result wynik operatora = rightR+leftR;
    QRES.push(wynikoperatora);
  }
}
  
```

Dla naszego zapytania wyglądałoby to następująco:





IMPLEMENTACJA





ZALICZENIE

Zaliczenie mini-projektu 2 polega na zaimplementowaniu diagramu UML pokazanego powyżej i zasymulowaniu w kodzie (np. w `main()`) poprzez sekwencję komend `pop` i `push` na stosie QRES, wykonania następujących operacji:

Dla osób z nazwiskami od A do J:

```
bag(1, 2.1, 3+4, "test" as nazwa);
```

Dla osób z nazwiskami od K do P:

```
struct(bag("ala", "ma", "kota"), 8*10, false);
```

Dla osób z nazwiskami od R do Z:

```
(bag("JPS" + "rules", 2.2, true)) groupas x;
```

Każda klasa rezultatu powinna być wyposażona w metodę `toString()` lub inną pozwalającą wypisać na ekran ostateczny rezultat zapytania (odwzorowując pełną strukturę rezultatu, np.

```
bag(0="Ala", 1="ma", 2=binder(name="co", value="kota"))
```

co byłoby rezultatem dla zapytania:

```
bag("Ala", "ma", "kota" as co);
```