Задачи за задължителна самоподготовка #6

по

Структури от данни и програмиране

email: kalin@fmi.uni-sofia.bg

9 януари 2015 г.

1. Да се дефинират `HashMap::operator [] (int)` и метод `HashMap::countKeys()`, които дават съответно:

   - i-тият пореден ключ, за който има записана стойност в хеш-таблицата. Подредбата на ключовете е без значение.
   - броят *различни* ключове, за които има записани стойности в хеш-таблицата.

2. Да се дефинира `HashTable HastTable::operator + (const HastTable&)`. Хеш-таблицата c, такава че $c = a + b$, да съдържа симетричната разлика на ключовете на a и b, със съответните им стойности. Хеш-функцията на c да е същата като на b.

   Симетрична разлика на множествата A и B наричаме множеството $C = A \triangle B = A \cup B - A \cap B$, съдържащо тези елементи на A, които не са елементи на B и тези елементи на B, които не са елементи на A.

3. Да се дефинира `HashTable HastTable::operator * (const HastTable&)`. Хеш-таблицата c, такава че $c = a * b$, да съдържа сечението на ключовете в a и b, със съответните им стойности от хеш-таблицата b. Хеш-функцията на c да е същата като на b.

4. Да се дефинира метод `HashTable::resize (int newSize)`. Методът да променя капацитета на хеш-таблицата като запазва всички ключове и съответните им стойности.

Обърнете внимание: Тъй като хеш-кодът на всеки ключ зависи от капацитета на хеш-таблицата, се налага т.нар. "рехеширане" - т.е. повторно изчисляване на хеш-кодове на всички ключове.

5. Йерархията на разработения на лекции интерпретатор да се обогати с виртуалния метод `Exporession::prettyPrint(ostream&)`. Методът да извежда програмата, съответна на даденото абстрактно синтактично дърво, спазвайки следните условия:

   - Всеки подизраз на даден израз да е на нов ред
   - Всеки подизраз на даден израз да е табулиран

   Например, следната програма:

   ```
   begin set sum 0 for i from 0 to 10 step 1 do begin print $i
   set sum + $sum $i end print $sum end
   ```

   Да се печата така:

   ```
   begin
       set sum 0
       for i from 0 to 10 step 1 do
           begin
               print $i
               set sum + $sum $i
           end
       print $sum
   end
   ```

   Можете да добавите всякакви допълнителни условия по ваше усмотрение.

6. Към предишната задача да се добави условието всеки ред на програмата да е номериран. Например:

   ```
   1: begin
   2:     set sum 0
   3:     for i from 0 to 10 step 1 do
   4:         begin
   5:             print $i
   6:             set sum + $sum $i
   7:         end
   8:     print $sum
   9: end
   ```

7. Йерархията на разработения на лекции интерпретатор да се обогати с виртуалния метод `Exporession::uniqueVariables()`. Методът да намира броя на различните променливи, използвани в програмата. За примера от предходната задача този брой е 2 - променливите sum и i.

8. Разработеният на лекции интерпретатор да се обогати с поддръжка на оператора $while$.