Семинар 16 октября

Задачи попроще

1 (с прошлого семинара)**.** N-граммой строки называется ее подстрока длины n. Напишите функцию

```
void calcNGramms(const string& str, size_t n, map<string, int>* result)
```

которая бы считала n-граммы по заданной строке и записывала бы в `*result` все n-граммы текста с указанием того, сколько раз такая n-грамма встретилась.

2. Реализуйте свой класс `Stack` на основе вектора. Предусмотрите в классе конструкторы и оператор присваивания, а также функции `empty`, `push`, `pop` и `top`.

3. Создайте класс `Counter`, представляющий простой счетчик. Предусмотрите операции для сброса счетчика и для преобразования его к целому числу. Перегрузите префиксный и постфиксный операторы `++`. Добавьте оператор присваивания, получающий на вход другой счетчик или целое число. Убедитесь, что с ним возможны констукции такого рода: `a = b = c`.

4. Напишите стековый калькулятор для выражений, записанных в постфиксной нотации. Например, `7 2 3 * -` эквивалентно `7 - 2 * 3`. Используйте `std::stack`.

Задачи посложнее

1. Напишите функцию

```
void makeIndex(
  const vector<string>& lines,
  map<string, vector<pair<int,int> > >* result
);
```

которая для каждого слова текста записывала бы в контейнер `result` все координаты (номер строки и начальная позиция в строке), по которым это слово встречается. Словом считается последовательность латинских букв.

2. Реализуйте класс `Queue` на основе двух векторов. Предусмотрите в классе конструкторы и оператор присваивания, а также функции `empty`, `push`, `pop` и `front`.

3. Реализуйте класс `Rational`, представляющий рациональные числа. Предусмотрите в классе конструкторы. Перегрузите операторы `+ - * / = == !=`, операторы сравнения, а также унарный минус.

4. Задан алфавит из открывающих и закрывающих символов, например: `() {} <> []`. Напишите функцию, которая проверяла бы, что строка, состоящая из этих символов, корректно сформирована. Используйте `std::stack`.