

```
1: // Copyright 2017 Sam Pickell
2: #include <cstdlib>
3: #include <vector>
4: #include <ctime>
5: #include <map>
6: #include <string>
7: #include "MarkovModel.hpp"
8:
9: MarkovModel::MarkovModel(std::string text, int k) {
10:     unsigned int i;
11:     std::string kgram;
12:
13:     for (i = 0; i < text.size() - k; i++) {
14:         kgram = text.substr(i, k);
15:         kgrams[kgram]++;
16:         // look for a new character. If found, add to our alphabet string
17:         std::size_t my_char = alphabet.find(text.at(i));
18:         if (my_char == std::string::npos) {
19:             alphabet.push_back(text.at(i));
20:         }
21:         kgram.push_back(text.at(i+k));
22:         kgrams[kgram]++;
23:     }
24:     for (unsigned int j = i; j <= text.size() - 1; j++) {
25:         int l;
26:         kgram = text.substr(j, text.size() - j);
27:         l = k - (text.size() - j);
28:         kgram.append(text.substr(0, l));
29:         kgrams[kgram]++;
30:         // look for a new character. If found, add to our alphabet string
31:         std::size_t my_char = alphabet.find(text.at(i));
32:         if (my_char == std::string::npos) {
33:             alphabet.push_back(text.at(i));
34:         }
35:         kgram.push_back(text.at(l));
36:         kgrams[kgram]++;
37:     }
38:
39:     private_order = k;
40: }
41:
42: MarkovModel::~MarkovModel() {
43: }
44:
45: int MarkovModel::freq(std::string kgram) {
46:     if (kgram.size() != private_order) {
47:         throw std::runtime_error(
48:             "freq: kgram is not of length k!");
49:     } else {
50:         return kgrams[kgram];
51:     }
52: }
53:
54: int MarkovModel::freq(std::string kgram, char c) {
55:     if (kgram.size() != private_order) {
56:         throw std::runtime_error(
57:             "freq: kgram is not of length k!");
58:     } else {
59:         if (private_order == 0) {
60:             std::string adv_c;
61:             adv_c.push_back(c);
```

```
62:         return kgrams[adv_c];
63:     } else {
64:         kgram.push_back(c);
65:         return kgrams[kgram];
66:     }
67: }
68: }
69:
70: char MarkovModel::randk(std::string kgram) {
71:     if (kgram.size() != private_order) {
72:         throw std::runtime_error(
73:             "randk: kgram is not of length k!");
74:     } else if (!kgrams[kgram]) {
75:         throw std::runtime_error(
76:             "randk: kgram does not exist!");
77:     } else {
78:         std::vector<char> my_vector;
79:         srand(time(NULL));
80:         unsigned int random_var = (rand() % freq(kgram)); // NOLINT
81:         for (unsigned int i = 0; i < alphabet.size(); i++) {
82:             int my_freq = freq(kgram, alphabet.at(i));
83:             if (my_freq >= 1) {
84:                 for (int j = 0; j < my_freq; j++) {
85:                     my_vector.push_back(alphabet.at(i));
86:                 }
87:             }
88:         }
89:         return my_vector.at(random_var);
90:     }
91: }
92:
93: std::string MarkovModel::gen(std::string kgram, int T) {
94:     std::string ret_string = kgram;
95:     int i = 0;
96:     while (ret_string.size() < (static_cast<unsigned int>(T))) {
97:         std::string temp;
98:         char c;
99:         temp = ret_string.substr(i, private_order);
100:        c = randk(temp);
101:        ret_string.push_back(c);
102:        i++;
103:    }
104:    return ret_string;
105: }
106:
107: std::ostream& operator<< (std::ostream &out, MarkovModel &mm) {
108:     for (std::map<std::string, int>::iterator p = mm.kgrams.begin();
109:          p != mm.kgrams.end(); p++) {
110:         out << p->first << '-' << p->second << std::endl;
111:     }
112:     out << "Order: " << mm.private_order << std::endl;
113:     out << "Alphabet: " << mm.alphabet << std::endl;
114:     return out;
115: }
```