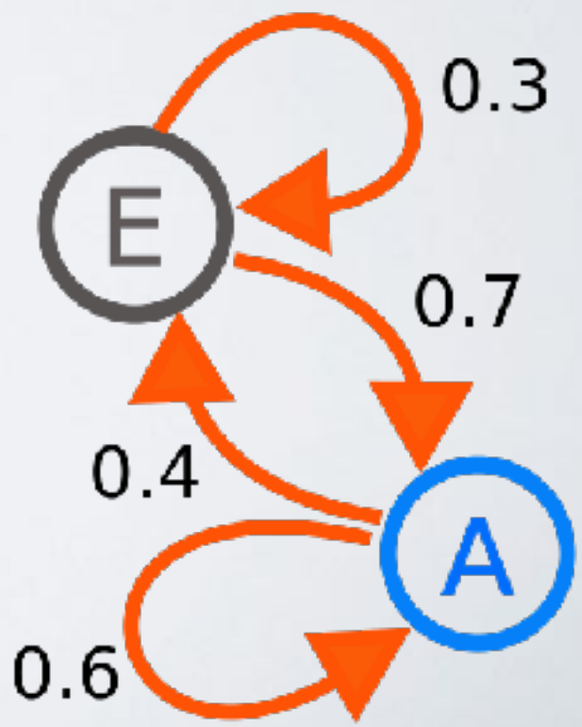
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Лекция № I / **12**
23.04.2018 г.

ЕЩЁ ОДНА ЗАДАЧА

Генерация случайного текста на основе цепей Маркова.

… Напомню, что большую часть программирования можно использовать идею friend классов настолько мало, что программист задает набор базовых классов, чтобы не следует использовать за исключением тех случаев, когда каждый из основных типов или как функцию без ассоциированных с ней статических данных. Если на три части: вначале получить ясное понимание задачи, потом выделить ключевые идеи, входящие в виде программы - именно в определение новых типов, не имело места. C++ в этом случае необходимости, усовершенствовать для английского - именно в программе…



- **Этап 1.** Чтение текста, построение таблицы переходов: **[слово$_1$, слово$_2$] → слово$_3$** (с учётом частот).
- **Этап 2.** Генерация.
  - Выбрать случайно два слова **w_1** и **w_2**.
  - Напечатать **w_1** и **w_2**.
  - В цикле:
    - Случайно выбрать **w_3** из слов, следующих за префиксом **w_1w_2** в тексте.
    - Напечатать **w_3**.
    - **$w_1, w_2 \leftarrow w_2, w_3$**.

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include "chain.h"

using namespace std;

int main(int argc, char **argv)
{
    if (argc != 3) {
        cerr << "Usage: markovka <text-file> <number-of-words>" << endl;
        return 1;
    }

    ifstream ifs(argv[1]);
    MarkovChain mc(ifs);

    mc.generate(cout, stoi(argv[2]));

    return 0;
}
```

```cpp
#pragma once

#include <istream>
#include <map>
#include <vector>
#include <string>

class MarkovChain {
public:
    MarkovChain(std::istream &is);

    void generate(std::ostream &os, int nwords);

private:
    // Префикс -> сколько всего раз префикс встречался
    std::map<std::vector<std::string>, size_t> prefix_counts;

    // Префикс -> вектор пар {слово за префиксом, сколько раз встр.}
    std::map<std::vector<std::string>,
             std::vector<std::pair<std::string, size_t> > > transitions;
};
```

```cpp
#include <algorithm>
#include <random>
#include <chrono>
#include <iterator>
#include <assert.h>
#include "chain.h"

using namespace std;

namespace {
    // Добавляет s к префиксу st. Полный размер префикса - p_size.
    void maintainPrefix(vector<string> &p, const string &s, int p_size) {
        if (p.size() < p_size)
            p.push_back(s);
        else {
            // Сдвиг влево на 1
            move(p.begin() + 1, p.end(), p.begin());
            p.back() = s;
        }
    }
}

// ...продолжение на следующем слайде
```

```
MarkovChain::MarkovChain(istream &is) {
    vector<string> prefix;
    string s;

    std::map<std::vector<std::string>, std::map<std::string, size_t> >
        all_counts;

    while (is >> s) {
        if (prefix.size() == 2) {
            prefix_counts[prefix]++;
            all_counts[prefix][s]++;
        }
        maintainPrefix(prefix, s, 2);
    }

    for (const auto &pair: all_counts) {
        auto &t = transitions[pair.first];

        for (const auto &pair2: pair.second)
            t.push_back(pair2);
    }
}
// ...продолжение на следующем слайде
```

```
void MarkovChain::generate(ostream &os, int nwords) {
    random_device rd;
    default_random_engine rng(rd());

    if (!prefix_counts.size())
        return;

    int count = 0, prefix_no = uniform_int_distribution<int>
        (0, prefix_counts.size() - 1)(rng);

    auto it = prefix_counts.begin();
    advance(it, prefix_no);

    vector<string> prefix = it->first;

    for (const auto &s : prefix) {
        if (count >= nwords)
            break;

        os << s << ' ';
        count++;
    }

    // ...продолжение на следующем слайде
```

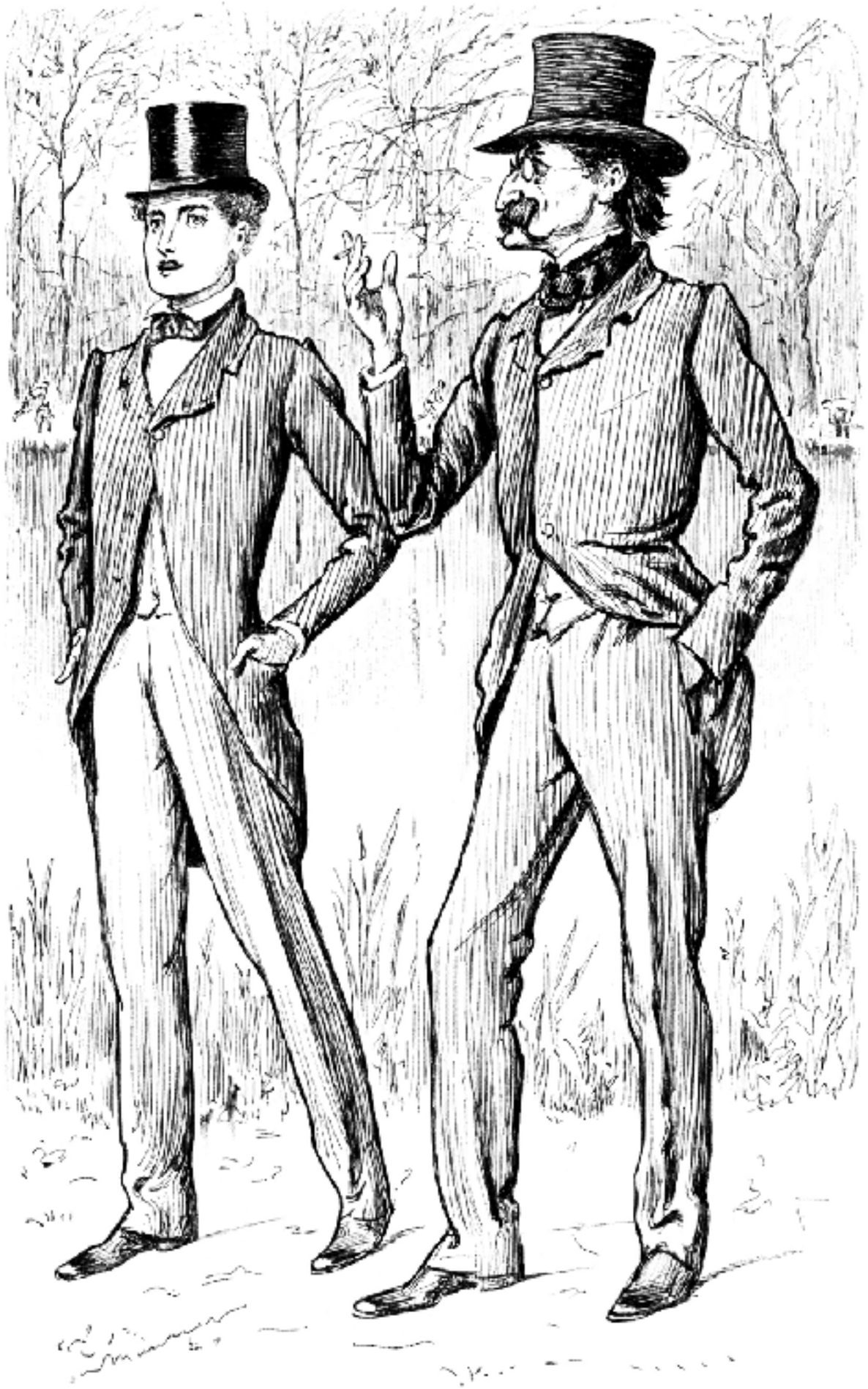
```
    for (; count < nwords; ++count) {
        const auto &options = transitions.at(prefix);
        size_t total_count = prefix_counts.at(prefix);
        size_t q = uniform_int_distribution<size_t>(0, total_count - 1)
                (rng);
        size_t accum = 0;
        string next_word;

        for (const auto &option : options) {
            if (q >= accum && q < accum + option.second) {
                next_word = option.first;
                break;
            }

            accum += option.second;
        }

        assert(!next_word.empty());
        os << next_word << ' ';

        maintainPrefix(prefix, next_word, 2);
    }
}
```

KISS

Keep. It. Simple. Stupid.

U. S. Navy, 1960

Принцип YAGNI



— Голубчик, что значит ЗСПВКБР?

— Сделайте самую простую вещь, которая будет работать!

И ЕЩЁ ОДНА ЗАДАЧА

Чтение и парсинг CSV-файлов.

```
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1996,Jeep,Grand Cherokee,"MUST SELL! air, moon roof, loaded",4799.00
```

Разделитель запятая

```
1965;Пиксел;E240 – формальдегид (опасный консервант)!;"красный, зелёный, битый";3000,00
1965;Мышка;"А правильней ""Использовать Ёлочки""";;4900,00
"Н/д";Кнопка;Сочетания клавиш;"MUST USE! Ctrl, Alt, Shift";4799,00
```

Разделитель точка с запятой

```
Year,Make,Model
1997,Ford,E350
2000,Mercury,Cougar
```

Заголовки

КОНЕЦ ДВЕНАДЦАТОЙ ЛЕКЦИИ