Министерство образования и науки Российской Федерации
Новосибирский государственный технический университет
Кафедра прикладной математики

Лабораторная работа №1
по дисциплине «Сравнительный анализ языков программирования»

Работа с STL C++. Исследование сложности алгоритма

Факультет	ПМИ
Группа	ПМ 33, ПМ 32
Студенты	Жиляков А.
	Антоненко В.
	Губский Е.
Преподаватели	Рояк М. Э.
	Ступаков И. М.
Вариант	5

Новосибирск, 2016

I Указания к выполнению работы

Разработайте программу для решения поставленной задачи на языке **C++**, удовлетворяющую следующим требованиям:

1. программа должна использовать для ввода-вывода потоки **STL**,
2. программа не должна содержать собственных реализаций стандартных алгоритмов и структур данных, а использовать существующие в **STL**,
3. размер каждой подпрограммы не должен превышать 10 строк.

Протестируйте разработанную программу. Исследуйте асимптотические свойства разработанной программы на системе тестов с возрастающей размерностью.
(`http://dispace.edu.nstu.ru/didesk/course/show/5626/5`)

I.1 Вариант задания

Задан большой текст (книга). Для наиболее часто встречающегося слова найти N наиболее часто встречающихся пар слов.

II Размышления

Для поиска наиболее популярного слова (или пары слов) удобно использовать контейнер std::map — ассоциативный массив с красно-чёрным деревом под капюшоном. Ключ суть слово, значение — частота.

Через n обозначим количество слов в нашей входной книжке. Тогда для занесения в контейнер данных потребуется $O(n \log m)$ времени (поиск в дереве осуществляется за логарифм), где $m \leq n$ — количество слов без повторений.

Положим $m = n$ (в такой книжке каждое слово уникально), чтобы проверить на практике логарифмическую сложность — в действительности $m \ll n$ и алгоритм работает практически за линейное время. Будем увеличивать длину книжки в 4 раза при начальном $n = 250000$. Тогда отношение времени работы программы суть

$$\frac{4n \log_2 4n}{n \log_2 n} = 4 \left(1 + \frac{2}{\log_2 n}\right) \searrow 4,\ n \to \infty, \tag{1}$$

то есть предполагается, что программа станет работать медленнее в 4+ раз, при этом при $n \to \infty$ отношение (1) должно становиться всё меньше и меньше.

III Тестирование

III.1 Тестирование сложности и проверка «размышлений»

i	n	t, сек	t_i/t_{i-1}
1	250 000	.742	
2	1 000 000	3.855	5.2
3	4 000 000	19.738	5.12
4	16 000 000	94.566	4.79

Таблица 1: Результаты тестирования при $m = n$

III.2 Тестирование на реальной книжке

частота	пара слов
65	martin eden
44	and martin
42	martin was
40	martin had
27	that martin
25	martin said
20	but martin
19	to martin
18	martin did
18	martin and
17	you martin
13	when martin
13	martin answered
13	as martin
12	martin went

Таблица 2: “Martin Eden,” Jack London. Число слов — 138 754

IV Приложение

Компилируем исходники с помощью MSVC:
`cl -EHsc -Ox testGen.cpp user.cpp`,
генерируем тест из n разных слов (n подаётся на стандартный ввод):
`testGen > dummy.txt`
и наша программа готова к работе:
`user < dummy.txt » out.txt`.

IV.1 testGen.cpp

```cpp
#include <random>
#include <algorithm>
#include <iterator>
#include <iostream>

int main() {
  int n;
  std::cin >> n;
  std::vector<int> v(n);
  for (int i = 0; i < n; ++i)
    v[i] = i;
    std::random_device rd;
    std::mt19937 g(rd());
    std::shuffle(v.begin(), v.end(), g);
    std::copy(v.begin(), v.end(), std::ostream_iterator<int>(std::cout, " "));
    std::cout << "\n";
  return 0;
}
```

IV.2 user.cpp

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <set>
#include <string>
#include <algorithm>
#include <ctime> //замерить время
#include <cctype> //isalnum

using namespace std; //программа коротенькая, так что можно и расслабиться

//сегмент данных

vector<string> book; //текст книги
string word; //самое популярное слово
set<string> words2ignore; //игнорируемые слова (предлоги, артикли и т.д.)
unsigned N = 15; //количество выводимых сочетаний слов

```

```
20
21  void getBook() { //считываем текст книжки из стандартного ввода в контейнер
22    while (cin >> word) {
23      transform(word.begin(), word.end(), word.begin(), ::tolower); //прописные буквы к строчным
24      word.erase(remove_if(word.begin(), word.end(), [](char c){return !isalnum(c);}), word.end()); //избавляемся от лишних символов в слове
25      if (word.length()) book.push_back(word);
26    }
27  }
28
29  void getWord() { //находим самое популярное слово
30    map<string, size_t> freq; //ключ - слово, значение - частота появления
31    size_t i, maxFreq = 0; //максимальная частота появления слова
32    for (i = 0; i < book.size(); ++i)
33      if (!words2ignore.count(book[i]) && ++freq[book[i]] > maxFreq) {
34        word = book[i];
35        maxFreq = freq[word];
36      }
37  }
38
39  int main() {
40    map<string, size_t> freq; //ключ - сочетание слов, значение - частота появления
41    map<string, size_t>::const_iterator iter;
42    vector<pair<size_t, string>> res; //отсортированный (см. далее) по частоте контейнер
43    size_t i;
44    clock_t begTime, endTime;
45    ifstream ignoreFile("ignore.txt");
46      while (ignoreFile >> word) words2ignore.insert(word);
47    getBook();
48
49      begTime = clock();
50      getWord();
51      for (i = 1; i < book.size() - 1; ++i)
52        if (book[i] == word) {
53          ++freq[book[i - 1] + " " + word];
54          ++freq[word + " " + book[i + 1]];
55        }
56      for (iter = freq.begin(); iter != freq.end(); ++iter)
57        res.push_back(make_pair(iter->second, iter->first));
58      sort(res.rbegin(), res.rend());
59      endTime = clock();
60      N = min(N, res.size());
61      for (i = 0; i < N; ++i)
62        cout << res[i].first << ' ' << res[i].second << '\n';
63      cout << "time diff: " << double(endTime - begTime) / CLOCKS_PER_SEC << '\n'
64         << "words count: " << book.size() << "\n\n";
65
66    return 0;
67  }
```