Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Факультет «О» Естественнонаучный

Кафедра О7 «Информационные системы и программная инженерия»

**Практическая работа №4**по дисциплине «Программирование на ЯВУ»  
на тему «STL»

Выполнил:  
Студент Костров Г. Ю. Группа О712Б  
Преподаватель: Васюков В.М.

Санкт-Петербург  
2022 г

**Постановка задачи:**

Дан файл с английскими словами, разделёнными знаком '#'. Используя std::getline или regex составить множество (std::set) из уникальных слов в файле. Слова хранить в std::string.  
Также составить словарь (std::map) по следующему принципу: Ключ - число, значение - количество слов из указанного числа букв в файле.  
Сгенерировать список из 100 структур со следующими полями:  
Название - случайное слово из множества.  
Основной параметр - используя std::advance получить значения трёх случайных элементов из словаря и перемножить их.  
Вторичный параметр - квадрат значения из словаря, связанного с названием(в случае буквы брать первую букву названия).  
Используя функции из библиотеки algorithms отсортировать список по возрастанию вторичного параметра.  
Используя std::copy\_if, создать новый список, содержащий только:  
Элементы с чётной разностью основного и вторичного параметра (округлить разность до целого).  
Сгенерировать на основе полученного отфильтрованного списка файл формата json.  
Добавить возможность загрузки файла формата json, создания на основе данных в нём списка, перетасовки (shuffle) его и отображения полученного результата.

**Данные из файла:**

generations#tyler#overall#committed#disorders#expires#georgia#parliamentary#sister#intel#frequencies#western#norway#spare#evening#mandatory#religious#subject#celtic#georgia#solid#tyler#humanities#original#indie#drug#toner#layout#western#holy#options#whore#occurs#healing#column#alerts#optical#atlanta#generations#reduction#possibilities#unable#options#simpsons#western#optical#equations#parenting#column#overall#quantities#toner#reduction#tournaments#agree#coleman#parliamentary#justin#penny#rental#position#holy#donald#ratio#shopping#sorts#dimension#seats#shoppercom#expires#parenting#lesbians#evening#alerts#beat#possibilities#wins#implement#alerts#subject#tyler#trained#touring#holy#optical#levitra#generations#integrating#wales#reviews#kits#hdtv#animated#column#operational#tyler#hash#donald#fundamentals#drug#parenting#norway#humanities#reviews#kilometers#penny#rental#quantities#ownership#wales#animated#mario#feeds#army#agree#rats#proteins#position#approx#trips#wales#feeds#dimension#parenting#drug#dimension#occurs#configuration#indie#evening#beach#trans#alphabetical#reduction#unable#implement#justin#donald#belong#touring#animated#generations#options#trips#left#justin#kilometers#integrating#equations#tournaments#implement#configuration#possibilities#drug#correlation#dimension#atlanta#perth#quantities#valentine#implement#norway#belong#shopping#planet#animated#wales#shopping#holy#ratio#original#ownership#rental#whore#justin#motels#spare#spare#integrating#reduction#touring#touring#subject#lesbians#intel#certified#feeds#quantities#evening#georgia#tournaments#lesbians#half#sister#alerts#equations#reviews#kilometers#indie#landing#perth#simpsons#layout#western#coleman#seats#left#position#proteins#sister#trained#whore#religious#unable#wins#shopping#trips#lesbians#perth#wins#indie#valentine#stakeholders#atlanta#seats#handjob#intel#hormone#norway#channels#sister#unable#verde#ownership#expires#intel

**Текст программы:**

**// Main.h**

#include <iostream>

#include <fstream>

#include <set>

#include <map>

#include <string>

#include <list>

#include <time.h>

#include <algorithm>

#include ".\single\_include\nlohmann\json.hpp"

#include <chrono>

#include <random>

#include "string\_view"

#include "cstring"

#include "..\helpFunctions\functions.cpp"

#include <iomanip>

using namespace std;

struct element

{

string name;

int first;

int second;

};

map<int,int> dictionary;

string dataFromFile;

set<string\_view> uniqueElements;

list<element> List;

list<element> sortedList;

list<element> copiedList;

vector<nlohmann::json> JSON;

vector<element> masFromJson;

void createSetAndMap( string\_view input\_string, const char separator);

void generatorStructures();

void sortirovka();

list<element> copyIf();

void createJson();

void fromJson();

template <typename T>

void printInfo(T container);

**// Main.cpp**

#include "main.h"

int main ()

{

ifstream file("data.txt");

getline(file,dataFromFile);

file.close();

createSetAndMap(dataFromFile, '#');

generatorStructures();

sortirovka();

copiedList = copyIf();

createJson();

fromJson();

int variant = 0;

do {

system("cls"); // очищаем экран

string menu[] = {

"What do you want to do?",

"1. Create set",

"2. Create map",

"3. Create 100 structures",

"4. Sort list of structures",

"5. Copy list with new parameters",

"6. Create new list in json",

"7. Json file input"};

printMenu(menu, 8);

exitMenu(8);

variant = getVariant(8);

system("cls"); // очищаем экран

switch (variant) {

case 1: {

// вывод множества

for(auto item : uniqueElements) {

cout << item << endl;

}

break;

}

case 2: {

// вывод словаря

for (auto i : dictionary) {

cout << "Key " << i.first << ", value " << i.second << endl;

}

break;

}

case 3: {

// вывод структур

printInfo(List);

break;

}

case 4: {

printInfo(sortedList);

break;

}

case 5: {

printInfo(copiedList);

break;

}

case 6: {

for(auto i : JSON){cout << i <<endl;}

break;

}

case 7: {

printInfo(masFromJson);

break;

}

}

system("pause");

} while (variant != 8);

}

// Добавление элемента в множество

void createSetAndMap( string\_view input\_string, const char separator)

{

size\_t part\_length = 0;

while( ( part\_length = input\_string.find( separator ) ) != input\_string.npos ) {

// set

uniqueElements.insert(input\_string.substr(0,part\_length));

// map

if( dictionary.find( part\_length ) == dictionary.end() ) {

dictionary.insert( make\_pair( part\_length, 1 ) );

} else {

auto i = dictionary.find( part\_length );

i->second++;

}

input\_string.remove\_prefix( part\_length + 1 );

}

if( !input\_string.empty() ) {

// set

uniqueElements.insert(input\_string.data());

// map

int lastWord = input\_string.length();

if( dictionary.find( lastWord ) == dictionary.end() ) {

dictionary.insert( make\_pair( lastWord, 1 ) );

} else {

auto i = dictionary.find( lastWord );

i->second++;

}

}

}

int Random(int left, int right) {

return rand()%right + left;

}

// Генератор структур

void generatorStructures()

{

for(int j=0;j<100;j++)

{

element temp = {"", 1, NULL}; // элемента списка

auto i = uniqueElements.begin();

advance(i, Random(0,26));

temp.name = \*i; // название элемента

// Основной параметр

for ( int m = 0; m < 3; m++) {

auto x = dictionary.begin();

advance(x,Random(0, dictionary.size()));

temp.first \*= x->second;

}

// Вторичный параметр

temp.second = dictionary[temp.name.length()] \* dictionary[temp.name.length()];

List.push\_back(temp);

}

}

// Сортировка списка

void sortirovka()

{

sortedList.assign(List.begin(), List.end());

sortedList.sort([](const element &a, const element &b){

return a.second < b.second;

});

}

// создание копии списка

list<element> copyIf() {

list<element> copiedList(List.size());

copy\_if(

List.begin(),

List.end(),

copiedList.begin(),

[](const element &a) {return ((a.first - a.second) % 2 == 0);}

);

copiedList.remove\_if([](const element &a) {return a.first+a.second == 0;});

return copiedList;

}

// запись в json

void createJson() {

nlohmann::json j{};

for(auto i : copiedList)

{

j["name"] = i.name;

j["first"] = i.first;

j["second"] = i.second;

JSON.push\_back(j);

}

ofstream file("new.json");

file << JSON;

file.close();

}

// открыть json

void fromJson()

{

ifstream file;

file.open("new.json");

nlohmann::json g{};

file >> g;

for(auto i : g)

{

element temp;

i.at("name").get\_to(temp.name);

i.at("first").get\_to(temp.first);

i.at("second").get\_to(temp.second);

masFromJson.push\_back(temp);

}

file.close();

shuffle(masFromJson.begin(),masFromJson.end(),default\_random\_engine(rand()));

}

template <typename T>

void printInfo(T container) {

cout << "| N | Name | First | Second |" << endl;

cout << "------------------------------------------------" << endl;

for(int N = 1; auto item : container) {

cout << "|" << setw(4) << N++ << " | ";

cout << setw(17) << item.name << " | ";

cout << setw(7) << item.first << " | ";

cout << setw(8) << item.second << " | " << endl;

};

cout << "------------------------------------------------" << endl;

}

**Результат работы программы:**

Рисунок 1: Основное меню.

Рисунок 2: Создание множества элементов.

Рисунок 3: Создание словаря.

Рисунок 4: Создание структур.

Рисунок 5: Сортировка структур.

Рисунок 6: Копирование по признаку.

Рисунок 7: Создание JSON файла.

Рисунок 8: Чтение JSON файла.

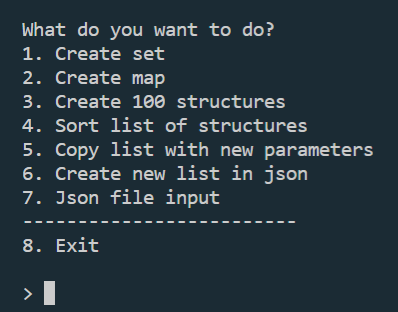


Рисунок 1

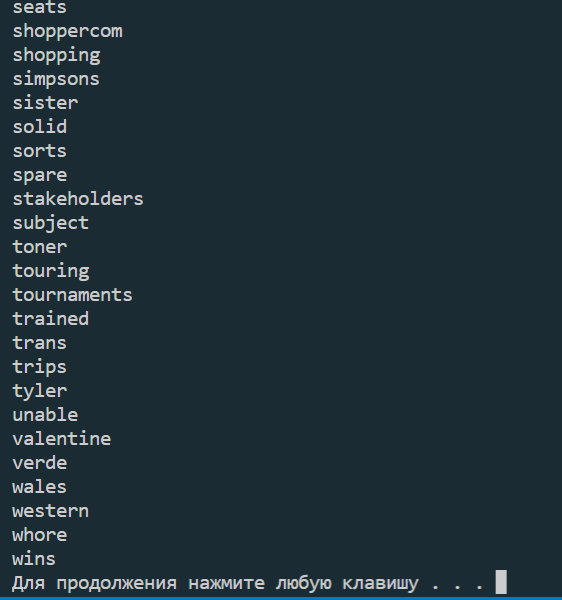


Рисунок 2

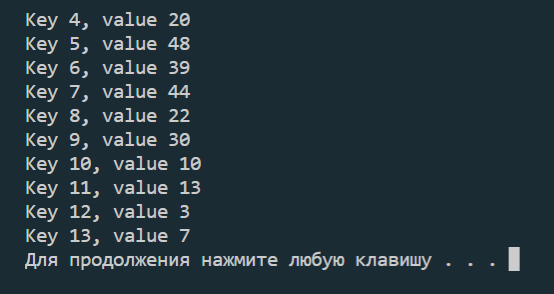


Рисунок 3

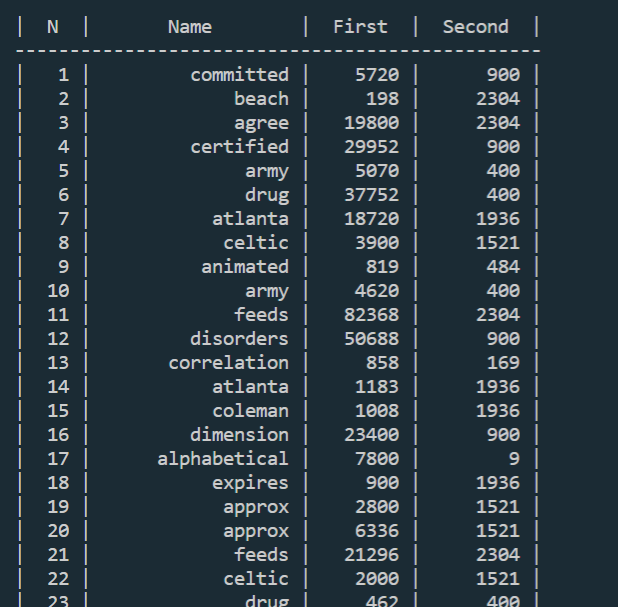


Рисунок 4

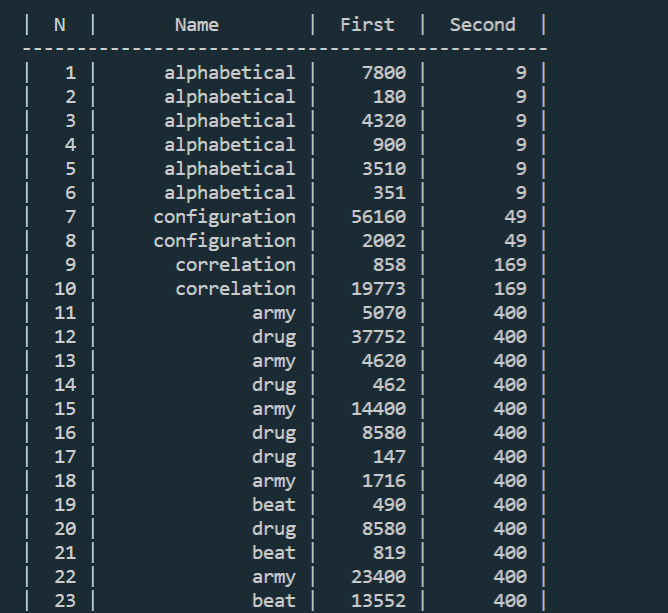


Рисунок 5

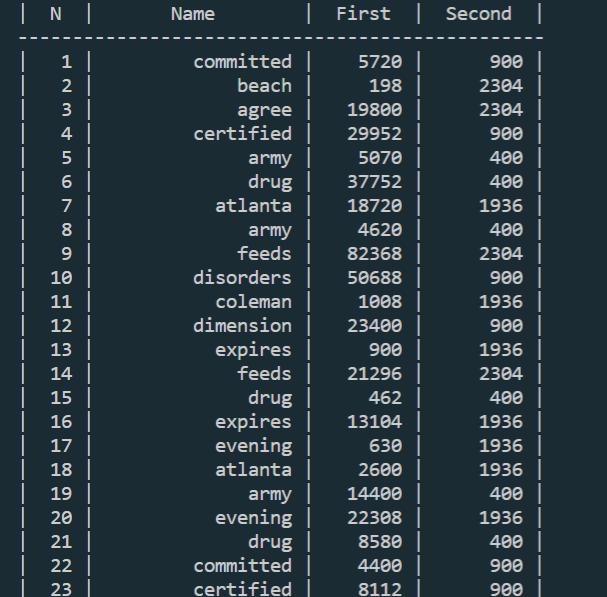


Рисунок 6

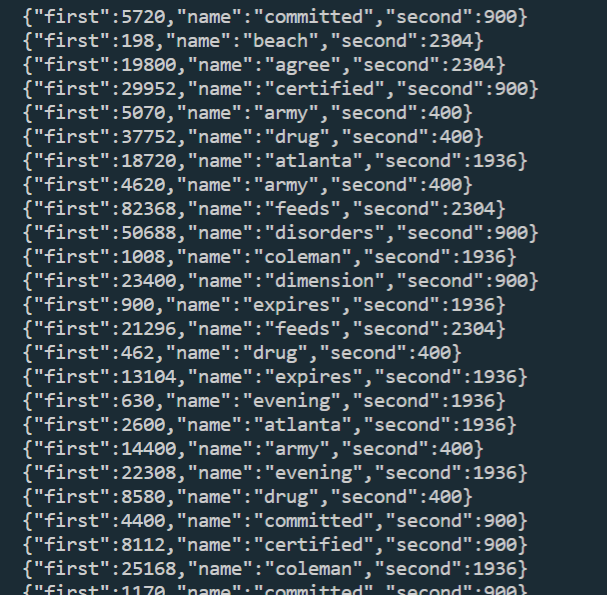


Рисунок 7

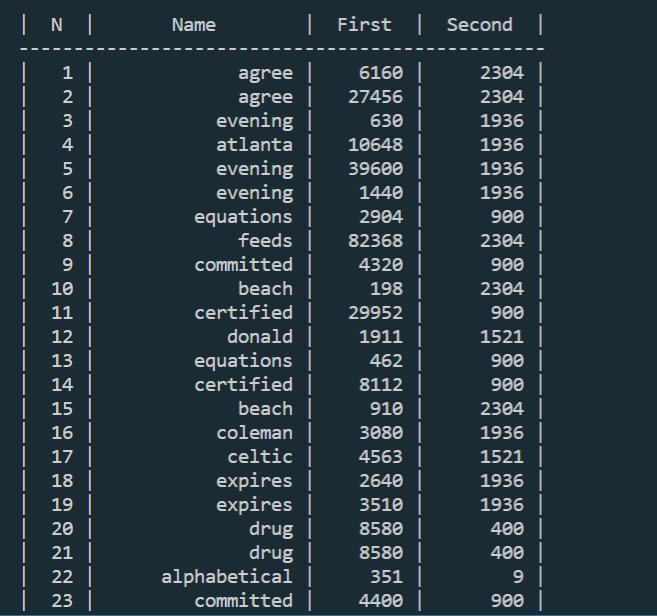


Рисунок 8