

Министерство высшего образования и науки РФ  
ФГБОУ ВО ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
Кафедра «Информационная безопасность систем и технологий»

ОТЧЁТ  
о практической работе №2  
Использование стандартной библиотеки Си++

Дисциплина: Языки программирования

Группа: 18ПИ1

Выполнил: Асаян А.В.

Количество баллов:

Дата сдачи:

Проверил: к.т.н., доцент Лупанов М.Ю.

Пенза, 2019

## 1 Цель работы

1.1 Знакомство с возможностями и использование стандартной библиотеки Си++.

## 2 Задания к практической работе

2.1 Написать программу, получающую значение текущего времени и выводящего его в том же формате, что и функция `ctime`, но русскими названиями дней недели и месяцев.

2.2 Написать программу, формирующую три целочисленных вектора размером 10000000 элементов и заполнить их случайными значениями в диапазоне от -1000000000 до 1000000000. Первый вектор должен создаваться пустым и элементы должны добавляться в цикле в конец вектора. Вторым вектор должен создаваться сразу нужного размера и заполняться с помощью алгоритма `generate`. Третий вектор должен быть создан как копия второго.

2.3 Используя возможности библиотеки `chrono` добавьте в предыдущую программу код для определения времени создания и заполнения каждого из трех векторов. Соберите программу в конфигурации «Release» и выполните несколько раз. Поясните полученные результаты замеров времени.

2.4 Добавить в предыдущую программу сортировку второго и третьего векторов. Один вектор отсортируйте с помощью алгоритма `sort`, второй с помощью алгоритма `stable_sort`. Добавьте код для определения времени сортировки и проведите эксперимент. Поясните полученные результаты замеров времени.

2.5 Разработать структуру данных для моделирования колоды карт (36 или 52, на ваш выбор). Написать код, выполняющий следующие действия: заполнение колоды, перемешивание колоды, поиск в колоде двух подряд карт одного цвета, поиск в колоде двух подряд карт одного номинала, поиск в колоде дамы пик, поиск в колоде всех тузов, печать колоды.

2.6 Разработать программу, читающую из файла список людей в формате «Фамилия Имя Отчество» и выполняющий с ним следующие действия:

сортировка по фамилии, поиск однофамильцев, поиск самого редкого имени(имен), поиск самого популярного имени (имен).

### 3 Результат выполнения работы

3.1 Была написана программа, получающая значение текущего времени и выводящая его в том же формате, что и функция `ctime`, но на русском языке. Была создана функция, получающая на вход указатель на объект структуры `tm`, он инициализируется значением текущего времени. В функции созданы два контейнера типа `vector`, хранящие значения типа `string`. Контейнеры содержат название месяцев и дней недели на русском языке. После чего, в зависимости от дня недели и месяца, которые хранятся в объекте структуры `tm` на экран выводятся их русскоязычный вариант. Результат работы программы представлен на рисунке 1.

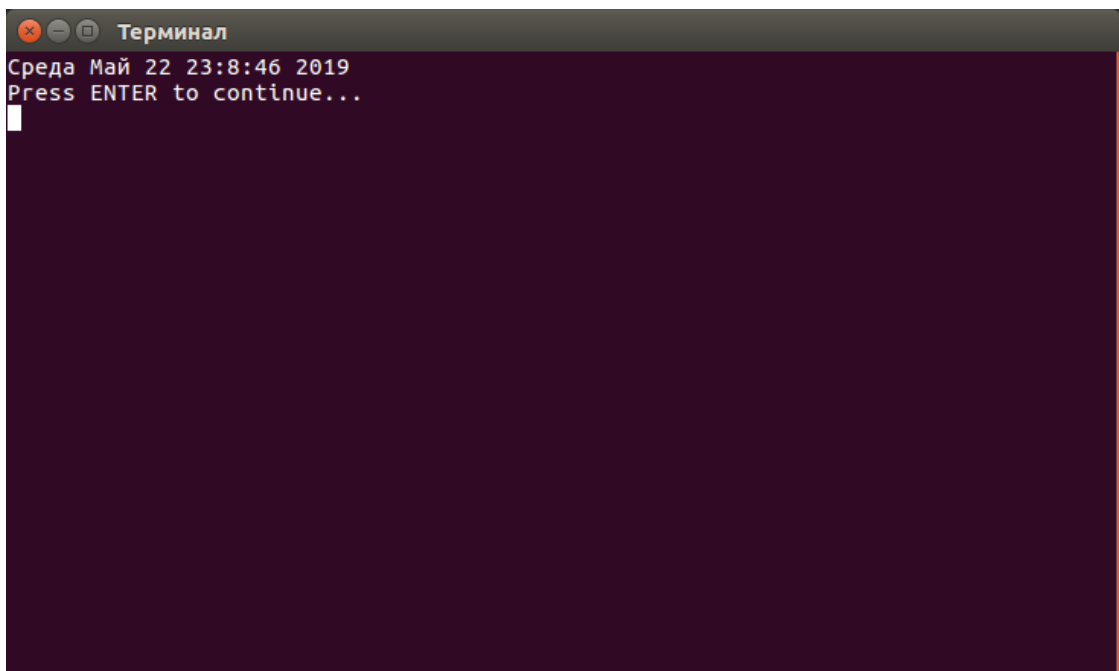


Рисунок 1 — Результат работы программы по выводу времени.

Код программы:

```
#include <iostream>
#include <ctime>
#include <string>
#include <vector>
using namespace std;
void Rustime(tm*ptm) {
```

```

        vector<string>wday{"Воскресенье", "Понедельник", "Вторник", "Среда", "Четверг", "Пятница", "Суббота"};
        vector<string>month{"Январь", "Февраль", "Март", "Апрель", "Май", "Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"};
        int i=ptm->tm_wday;
        int k=ptm->tm_mon;
        cout<<wday[i]<<"    "<<month[k]<<"    "<<ptm->tm_mday<<"    "<<ptm->tm_hour<<":"<<ptm->tm_min<<":"<<ptm->tm_sec<<"    "<<1900+ptm->tm_year<<endl;
    }
    int main(int argc, char **argv)
    {
        time_t t = time(NULL);
        tm*ptm;
        ptm=localtime(&t);
        Rustime(ptm);
        return 0;
    }

```

3.2 Была разработана программа, создающая 3 вектора целочисленного типа, первый вектор не инициализируется при создании, а после заполняется в цикле добавлением в конец, значениями, которые были сгенерированы функцией RandomGenerator(), пример которой был дан в методических указаниях. Второй вектор сразу создаётся необходимого размера и заполняется с помощью алгоритма generate. Третий вектор создаётся как копия второго. Код программы:

```

#include <iostream>
#include <vector>
#include <algorithm>
#include <random>
using namespace std;
int RandomGenerator()
{
    static mt19937 rnd((uint64_t)&rnd);
    uniform_int_distribution<int>d(1000000000,10000000000);
    return d(rnd);
}
int main(int argc, char **argv)
{
    int rd;

```

```

        vector<int> first;
        vector<int> second(10000000);
        for(int i=0;i<10000000;i++){
            rd=RandomGenerator();
            first.push_back(rd);
        }
        generate(second.begin(),second.end(),RandomGenerator());
        vector<int> third(second);
        return 0;
    }

```

3.3 При помощи возможностей, предоставляемых библиотекой `chrono`, была написана программа, измеряющая интервалы времени, которые нужны для создания каждого из трёх векторов в программе, написанной в пункте 3.2. Для измерения времени необходимо использовать `time_point` и метод `now()`, класса `steady_clock`. Результаты работы программы представлены на рисунке 2.

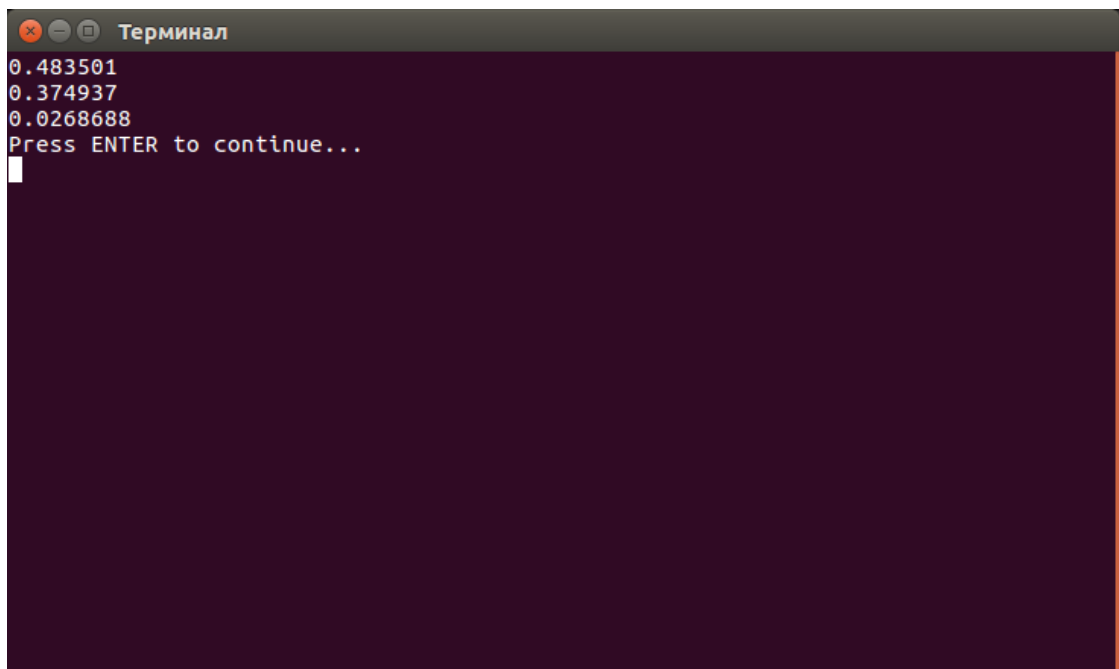


Рисунок 2 — Результаты измерения времени, необходимого для создания векторов.

По измерениям можно судить о том, что первый вектор создается и заполняется дольше, чем второй и третий, так как его размер не задан изначально и он заполняется в цикле добавлением значений в конец. Эти

значения в цикле сначала генерируются, после чего добавляются в массив. Размер второго вектора был задан изначально, сгенерированные значения сразу записываются в него. Третий вектор просто является копией второго, для него не генерируются новые значения, поэтому его создание занимает меньше времени.

3.4 В программу разработанную в пункте 3.2 и модифицированную в пункте 3.3 были добавлены алгоритмы сортировки `sort` и `stable_sort`. Так же были добавлены `time_point`, метод класса `steady_clock::now()`, объекты класса `duration`, в который записывается интервал времени необходимый для сортировки векторов. Код программы:

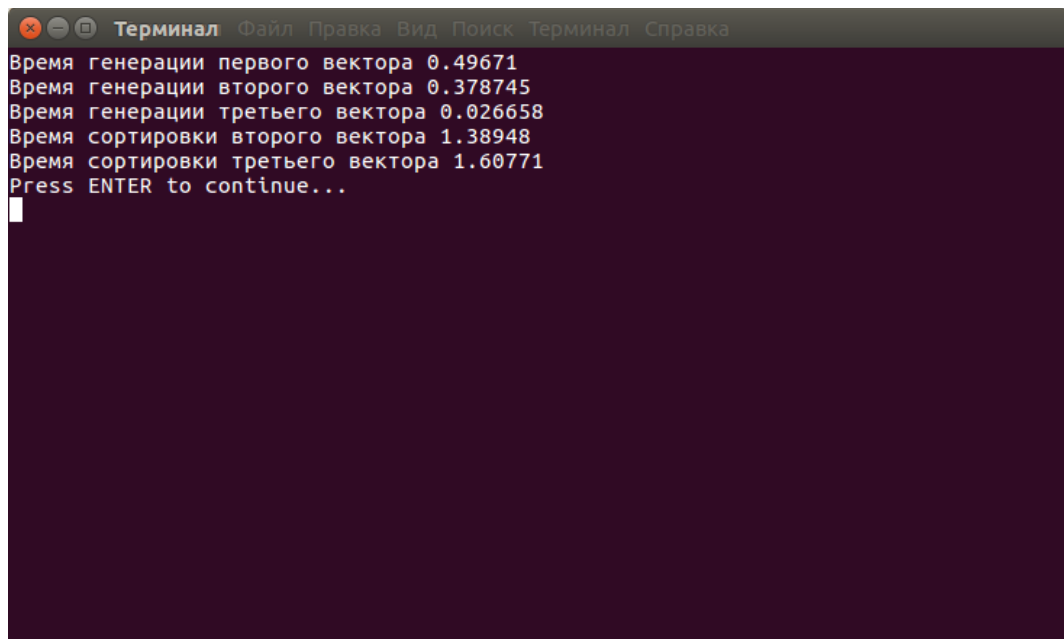
```
#include <iostream>
#include <vector>
#include <algorithm>
#include <random>
#include <chrono>
using namespace std;
using namespace std::chrono;
int RandomGenerator()
{
    static mt19937 rnd((uint64_t)&rnd);
    uniform_int_distribution<int> d(-
1000000000,1000000000);
    return d(rnd);
}
int main(int argc, char **argv)
{
    int rd;
    vector<int> first;
    steady_clock::time_point tp1f =
steady_clock::now();
    for(int i=0; i<10000000; i++) {
        rd=RandomGenerator();
        first.push_back(rd);
    }
    steady_clock::time_point tp2f =
steady_clock::now();
    duration<double> d1 = tp2f-tp1f;
```

```

        cout<<"Время генерации первого вектора
"<<d1.count()<<endl;
        vector<int> second(10000000);
        steady_clock::time_point tp1s =
steady_clock::now();
        generate(second.begin(),second.end(),RandomGenera
tor);
        steady_clock::time_point tp2s =
steady_clock::now();
        duration<double> d2 = tp2s-tp1s;
        cout<<"Время генерации второго вектора
"<<d2.count()<<endl;
        steady_clock::time_point tp1t =
steady_clock::now();
        vector<int> third(second);
        steady_clock::time_point tp2t =
steady_clock::now();
        duration<double> d3 = tp2t-tp1t;
        cout<<"Время генерации третьего вектора
"<<d3.count()<<endl;
        steady_clock::time_point tp1sr =
steady_clock::now();
        sort(second.begin(),second.end());
        steady_clock::time_point tp2sr =
steady_clock::now();
        duration<double> dsr = tp2sr-tp1sr;
        cout<<"Время сортировки второго вектора
"<<dsr.count()<<endl;
        steady_clock::time_point tp1ssr =
steady_clock::now();
        stable_sort(third.begin(),third.end());
        steady_clock::time_point tp2ssr =
steady_clock::now();
        duration<double> dssr = tp2ssr-tp1ssr;
        cout<<"Время сортировки третьего вектора
"<<dssr.count()<<endl;
        return 0;
    }

```

На рисунке 3 изображены результаты работы программы.

A screenshot of a terminal window with a dark background and light-colored text. The window title bar shows 'Терминал' (Terminal) and several menu items: 'Файл', 'Правка', 'Вид', 'Поиск', 'Терминал', and 'Справка'. The terminal output displays the following lines: 'Время генерации первого вектора 0.49671', 'Время генерации второго вектора 0.378745', 'Время генерации третьего вектора 0.026658', 'Время сортировки второго вектора 1.38948', 'Время сортировки третьего вектора 1.60771', and 'Press ENTER to continue...'. A cursor is visible on the line following the last message.

```
Терминал  Файл  Правка  Вид  Поиск  Терминал  Справка
Время генерации первого вектора 0.49671
Время генерации второго вектора 0.378745
Время генерации третьего вектора 0.026658
Время сортировки второго вектора 1.38948
Время сортировки третьего вектора 1.60771
Press ENTER to continue...
█
```

Рисунок 3 — Время сортировки векторов.

Как видно из измерений, сортировка вектора два заняла меньше времени, чем сортировка вектора три, так как были использованы разные алгоритмы сортировки. Алгоритм `sort` не гарантирует того, что порядок равных значений в контейнере окажется таким же, как и до сортировки, а алгоритм `stable_sort` оставляет порядок равных значений гарантировано останется неизменным.

3.5 Была создана структура моделирующая колоду из 36 карт. Атрибуты структуры содержат масть и значение карты. Колода перемешивается с помощью алгоритма `shuffle`, используя генератор вихрь Мерсена. Код программы:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <random>
#include <chrono>
using namespace std;
struct Cards {
    int suit;
    int value;
    Cards(int w,int h) {
        suit=w;
        value=h;
    }
};
```



```

    }
    friend ostream& operator<< (ostream &out, const
Cards &b) {
        out << "SUIT " << b.suit << ", VALUE " <<
b.value;
    }
    friend bool operator== (const Cards &a, const
Cards &b) {
        return (a.suit == b.suit and a.value==
b.value);
    }
};
int main(int argc, char **argv)
{
    enum suit {peaks=0,clubs=1,hearts=2,diamond=3};
                                enum          value
{six,seven,eight,nine,ten,jack,queen,king,ACE};
    Cards y(0,0);
    vector <Cards> deck;
    for(int k=0; k<4; k++) {
        for(int i=0; i<9; i++) {
            Cards y(k,i);
            deck.push_back(y);
        }
    }

    shuffle(deck.begin(),deck.end(),mt19937(time(NULL)
)));
    for(int i=0; i<36; i++)
        cout<<deck[i]<<endl;
    for(int i=0; i<36; i++) {
                                if((deck[i].suit==diamond    or
deck[i].suit==hearts)    and    (deck[i+1].suit==hearts    or
deck[i+1].suit==diamond)) {
                                    cout<<"Пара карт красной масти в позициях
"<<i+1<< ", "<<i+2<<endl;
                                    cout<<deck[i]<<" "<<deck[i+1]<<endl;
                                }
    }
    for(int i=0; i<36; i++) {
        if(deck[i].value==deck[i+1].value) {
            cout<<"Пара карт с одинаковыми значениями
в позициях "<<i+1<< ", "<<i+2<<endl;
            cout<<deck[i]<<" "<<deck[i+1]<<endl;
        }
    }
}

```

```

    }
}
Cards q(clubs, queen);
auto it1=find(deck.begin(), deck.end(), q);
auto index = distance(deck.begin(), it1);
    cout<<"Пиковая дама в колоде на "<<index+1<<"
позиция"<<endl;
    for(int i=0; i<36; i++) {
        if(deck[i].value==ACE) {
            cout<<"Туз ";
            cout<<deck[i]<<". Позиция в колоде
"<<i+1<<endl;
        }
    }
    return 0;
}

```

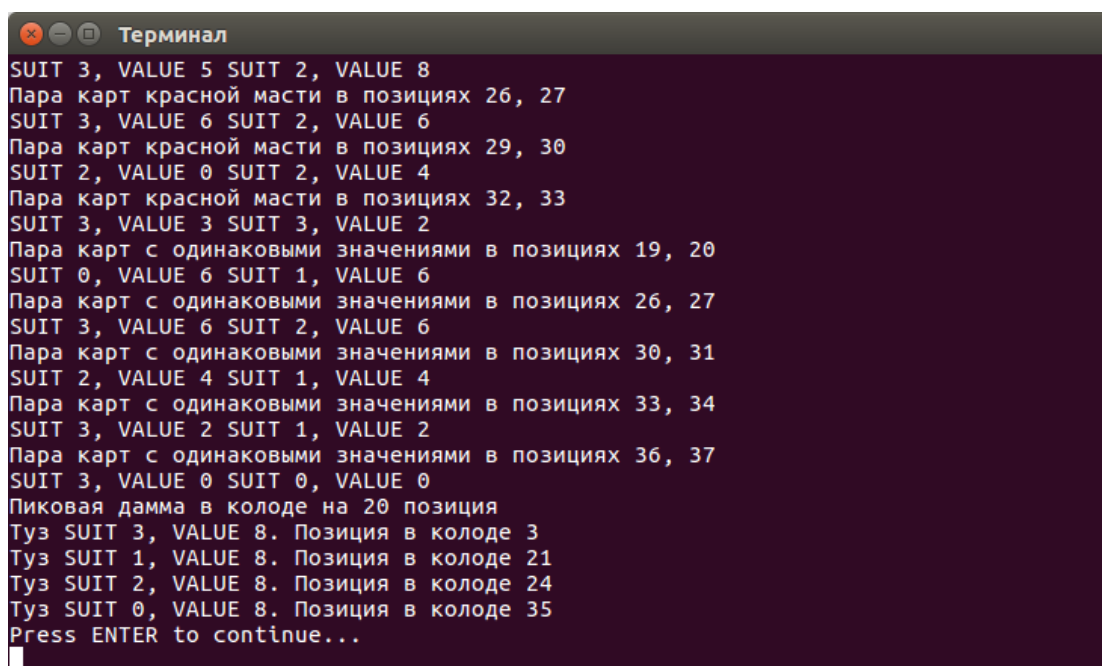
На рисунках 4 и 5 представлен результат работы программы.

```

Пара карт красной масти в позициях 3, 4
SUIT 3, VALUE 8 SUIT 2, VALUE 2
Пара карт красной масти в позициях 4, 5
SUIT 2, VALUE 2 SUIT 3, VALUE 1
Пара карт красной масти в позициях 16, 17
SUIT 2, VALUE 3 SUIT 2, VALUE 1
Пара карт красной масти в позициях 17, 18
SUIT 2, VALUE 1 SUIT 2, VALUE 7
Пара карт красной масти в позициях 23, 24
SUIT 3, VALUE 5 SUIT 2, VALUE 8
Пара карт красной масти в позициях 26, 27
SUIT 3, VALUE 6 SUIT 2, VALUE 6
Пара карт красной масти в позициях 29, 30
SUIT 2, VALUE 0 SUIT 2, VALUE 4
Пара карт красной масти в позициях 32, 33
SUIT 3, VALUE 3 SUIT 3, VALUE 2
Пара карт с одинаковыми значениями в позициях 19, 20
SUIT 0, VALUE 6 SUIT 1, VALUE 6
Пара карт с одинаковыми значениями в позициях 26, 27
SUIT 3, VALUE 6 SUIT 2, VALUE 6
Пара карт с одинаковыми значениями в позициях 30, 31
SUIT 2, VALUE 4 SUIT 1, VALUE 4
Пара карт с одинаковыми значениями в позициях 33, 34
SUIT 3, VALUE 2 SUIT 1, VALUE 2

```

Рисунок 4 — Результат работы программы «Колода»1.



```
Терминал
SUIT 3, VALUE 5 SUIT 2, VALUE 8
Пара карт красной масти в позициях 26, 27
SUIT 3, VALUE 6 SUIT 2, VALUE 6
Пара карт красной масти в позициях 29, 30
SUIT 2, VALUE 0 SUIT 2, VALUE 4
Пара карт красной масти в позициях 32, 33
SUIT 3, VALUE 3 SUIT 3, VALUE 2
Пара карт с одинаковыми значениями в позициях 19, 20
SUIT 0, VALUE 6 SUIT 1, VALUE 6
Пара карт с одинаковыми значениями в позициях 26, 27
SUIT 3, VALUE 6 SUIT 2, VALUE 6
Пара карт с одинаковыми значениями в позициях 30, 31
SUIT 2, VALUE 4 SUIT 1, VALUE 4
Пара карт с одинаковыми значениями в позициях 33, 34
SUIT 3, VALUE 2 SUIT 1, VALUE 2
Пара карт с одинаковыми значениями в позициях 36, 37
SUIT 3, VALUE 0 SUIT 0, VALUE 0
Пиковая дама в колоде на 20 позиция
Туз SUIT 3, VALUE 8. Позиция в колоде 3
Туз SUIT 1, VALUE 8. Позиция в колоде 21
Туз SUIT 2, VALUE 8. Позиция в колоде 24
Туз SUIT 0, VALUE 8. Позиция в колоде 35
Press ENTER to continue...
```

Рисунок 5 - Результат работы программы «Колода»2.

3.6 Была разработана программа, читающая список ФИО из файла и совершающая с ними следующие действия: сортировка по фамилии, поиск однофамильцев, поиск самого редкого имени(имен), поиск самого популярного имени (имен). Для хранения списка ФИО была создана структура, объекты которой записывались в вектор. Для сортировки списка был перегружен оператор < и использован алгоритм sort. Для поиска самого популярного имени был использован контейнер map<string,int>, в который записывались фамилии из сортированного списка. После его с помощью алгоритмов max\_element и min\_element находится самый популярный и самый редкий элемент. Код программы:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <algorithm>
#include <set>
#include <string>
#include <sstream>
using namespace std;
struct Data {
```

```

string FirstName;
string LastName;
string Patronymic;
//vector<int> Evaluations;
friend bool operator< (const Data &d1, const Data
&d2) {
    if(d1.FirstName != d2.FirstName) {
        return d1.FirstName < d2.FirstName;
    } else {
        if(d1.LastName != d2.LastName) {
            return d1.LastName < d2.LastName;
        } else {
            if(d1.Patronymic != d2.Patronymic) {
                return d1.Patronymic <
d2.Patronymic;
            }
        }
    }
}

friend ostream& operator<< (ostream &out, const
Data &data) {
    cout<<data.FirstName<<" "<<data.LastName<<"
"<<data.Patronymic<<endl;
}

};
int main()
{
    ifstream Read("name.txt");
    if (!Read) {
        cout << "Cannot open file" << endl;
        return 1;
    }
    int str=0;
    string line;
    while(getline(Read, line)) {
        stringstream lineStream(line);
        while(getline(lineStream, line, ' ')) {
            ++str;
        }
    }
    str=str/3;
    vector<Data> slova;
    Read.clear();

```

```

Read.seekg(ios::beg);
Data data;
for(int i=0; i<str; i++) {
    Read >> data.FirstName;
    Read >> data.LastName;
    Read >> data.Patronymic;
    slova.push_back(data);
}
cout<<"До сортировки: "<<endl;
for(int i=0; i<str; i++) {
    cout<<slova[i];

}
sort(slova.begin(), slova.end());
cout<<"После сортировки: "<<endl;
for(int i=0; i<str; i++) {
    cout<<slova[i];

}
int t;
for(int i=0; i<str; i++) {
    if(slova[i].FirstName==slova[i+1].FirstName)
{
        cout<<"Однофамильцы: "<<endl;
        t=i;
        string fn=slova[i].FirstName;
        while(slova[t].FirstName==fn) {

            cout<<slova[t];
            t++;
        }
        i=t-1;
    }
}
map<string, int> ounter;
map<string, int>::iterator it;
for(int i=0; i<str; i++) {
    ounter[slova[i].LastName]++;
}

    auto x = std::max_element(ounter.begin(),
ounter.end(), [](const pair< string, int>& p1, const
pair<string, int>& p2) {
    return p1.second < p2.second;
});
}

```

```

        auto y = std::min_element(ounter.begin(),
ounter.end(), [](const pair< string, int>& p1, const
pair<string, int>& p2) {
    return p1.second < p2.second;
});
cout<<"Most popular: "<<endl;
for(it=ounter.begin();it!=ounter.end();it++){
    if(it->second==x->second)
        cout<<it->first<<endl;
}
cout<<"Most rare: "<<endl;
for(it=ounter.begin();it!=ounter.end();it++){
    if(it->second==y->second)
        cout<<it->first<<endl;
}
Read.close();
}

```

На рисунках 6 и 7 представлены результаты работы программы.

```

Терминал Файл Правка Вид Поиск Терминал Справка
До сортировки:
Александров Иван Петрович
Гистапин Иван Васильевич
Александров Владимир Олегович
Ванин Сергей Константинович
Демидов Алексей Павлович
Антонов Вадим Олегович
Прохоров Иван Петрович
Петров Сергей Анатольевич
Александров Евгений Васильевич
Александров Владимир Николаевич
Демидов Валерий Павлович
Петров Евграф Евгеньевич
Петров Анатолий Никитич
Петров Сергей Павлович
Асаян Артём Вадимович
Яров Михаил Васильевич
Гасин Евграф Артёмович
После сортировки:
Александров Владимир Николаевич
Александров Владимир Олегович
Александров Евгений Васильевич
Александров Иван Петрович
Антонов Вадим Олегович

```

Рисунок 6 — Результаты работы программы по обработке списка ФИО1.

```
Терминал
После сортировки:
Александров Владимир Николаевич
Александров Владимир Олегович
Александров Евгений Васильевич
Александров Иван Петрович
Антонов Вадим Олегович
Асаян Артём Вадимович
Ванин Сергей Константинович
Гасин Евграф Артёмович
Гистапин Иван Васильевич
Демидов Алексей Павлович
Демидов Валерий Павлович
Петров Анатолий Никитич
Петров Евграф Евгеньевич
Петров Сергей Анатольевич
Петров Сергей Павлович
Прохоров Иван Петрович
Яров Михаил Васильевич
Однофамильцы:
Александров Владимир Николаевич
Александров Владимир Олегович
Александров Евгений Васильевич
Александров Иван Петрович
Однофамильцы:
```

Рисунок 7 - Результаты работы программы по обработке списка ФИО2.

```
Терминал
Однофамильцы:
Александров Владимир Николаевич
Александров Владимир Олегович
Александров Евгений Васильевич
Александров Иван Петрович
Однофамильцы:
Демидов Алексей Павлович
Демидов Валерий Павлович
Однофамильцы:
Петров Анатолий Никитич
Петров Евграф Евгеньевич
Петров Сергей Анатольевич
Петров Сергей Павлович
Most popular:
Иван
Сергей
Most rare:
Алексей
Анатолий
Артём
Вадим
Валерий
Евгений
Михаил
```

Рисунок 8 — Результаты работы программы по обработке списка ФИО2.

#### 4 Вывод

В результате практической работы были основаны возможности стандартной библиотеки C++. Были изучены контейнеры, возможности работы со временем, алгоритмы и генераторы псевдослучайных чисел. Были получены

практические навыки по применению контейнера `vector` и `map`, класса `steady_clock`, алгоритмов `sort`, `stable_sort`, генератора случайных чисел `mt19937`.