

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

ОТЧЕТ
о практической работе №2
ИСПОЛЬЗОВАНИЕ СТАНДАРТНОЙ БИБЛИОТЕКИ СИ++

Дисциплина: Языки программирования

Группа: 18ПИ1

Выполнил: Нестеров И.С.

Количество баллов:

Дата сдачи:

Принял: к.т.н., доцент Лупанов М. Ю.

Пенза 2019

1 Цель работы

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

2 Задание к лабораторной работе

2.1 Напишите 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 Напишите программу, формирующую три целочисленных вектора размером 10000000 элементов и заполните их случайными значениями в диапазоне от -1000000000 до 1000000000. Первый вектор должен создаваться пустым и элементы должны добавляться в цикле в конец вектора. Второй 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 Добавьте 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 Разработайте структуру данных для моделирования колоды карт (36 или 52, на ваш выбор). Напишите 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 Разработайте 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 Результаты работы

3.1 Чтобы функция `ctime` выводила значение текущего времени и названия дней недели и месяцев на русском языке, нужно использовать оператор `switch`, который обрабатывает и выводит полученные данные с `tm_wday`, дней недели от 0 до 6, и `tm_mon`, месяцев от 0 до 11. Пример работы программы представлен на рисунке 1. Полный текст представлен в приложении А.

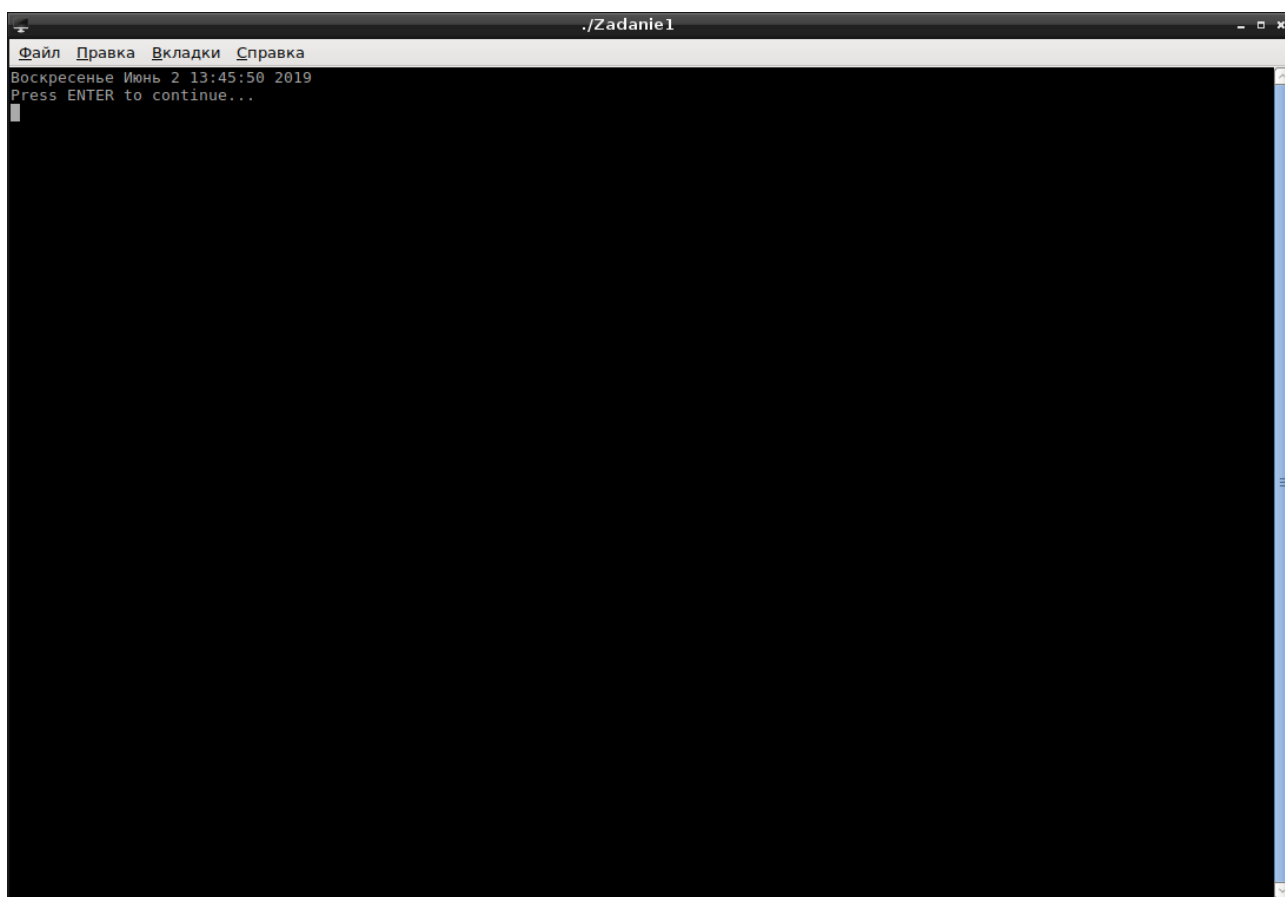


Рисунок 1 - Результат работы программы 1

3.2 Первый вектор создали пустым и элементы добавляются в цикле в конец вектора с помощью метода `push_back()`. Второй вектор создали сразу нужного размера, то есть на 10000000 элементов, и заполняли его с помощью

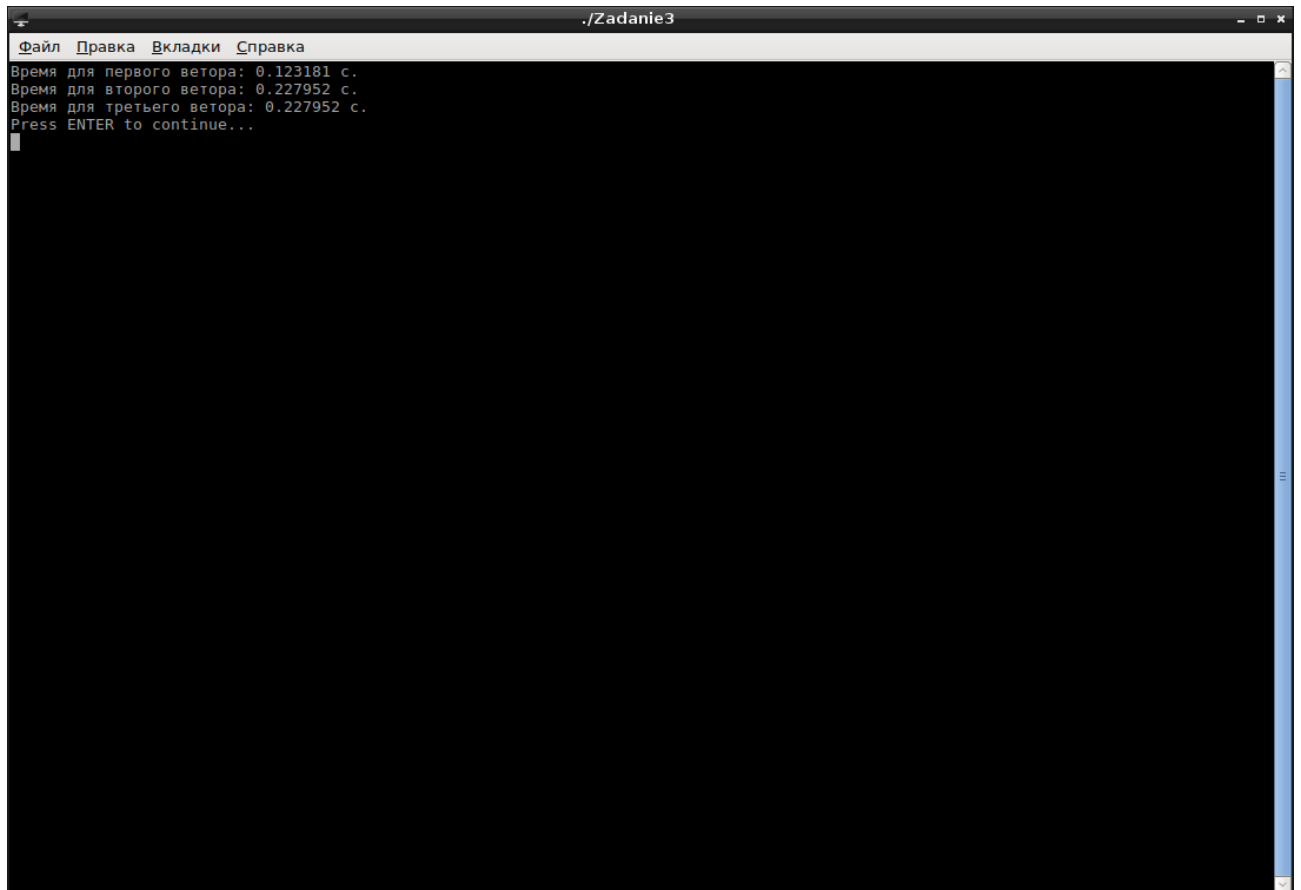
алгоритма generate. Третий вектор создали как копию второго. Пример работы программы представлен на рисунке 2. Полный текст представлен в приложении Б.

Файл	Правка	Вкладки	Справка							
998	-905929186	476401317	269377665	840653524	235765940	-213678811	594688459	-235386720	7	
62959626	310362502	648180530	351049317	915190275	652837359	-746644305	111646081	-85584		
0127	922159884	-458848283	-420357656	779515593	-231910682	189683833	175525439	605909919	-	
296961587	-26935033	669061903	-847906602	-918732699	216124262	631673101	224115355	-59602		
6994	770736347	-855175199	363430154	-862341613	-677418831	-434553733	439894375	-20747765	9	
46148995	-750305798	395698136	966489816	218977411	914734249	-446625697	-145145381	-11657		
8920	296695861	-539095629	-1072794	-42353138	-573424881	256146635	-47135458	659884927	3	
43369755	785438366	-890143761	977645121	-896378906	-340235944	301506220	238732885	-25057		
8662	290514984	369317054	-87053659	950138500	151306288	-696741570	-786351727	433775700	8	
3410243	-463253507	-243040742	-123021042	700703737	681512689	476052169	943678087	-535792087	-	
123509721	607268585	985404638	-679788564	-636430248	-353233626	-4482760	411906865	-20394		
2786	281579365	-16177998	821434808	834846850	-132094883	869874106	668799855	341588112	2	
35620102	-177841207	8405168	199039670	-579189858	-968917790	349659894	-21244867	722481544	-	
655794618	-260938663	260260964	-442973827	785154804	106604383	-110505776	296960336	-88661		
7100	-542097497	-992566798	-29742331	951161835	-106638220	440193777	-582807071	629901843	7	
08495136	-641698271	-936375950	-886137050	-85976765	860413603	9515653	-987026323	-119134819	5	
16753664	-298253555	253664914	-199099870	-529217567	-642875219	339762902	-650010010	653006		
749	-280828255	-340468046	42013404	165180093	568691588	-793199248	-639069443	-103643250	-	
565310668	52659721	611769505	-750820903	545290811	-682192285	-267224769	918824898	238766		
377	213734787	-202421810	-562281667	60596761	-520652369	-7887252	-936575191	30700194	-	
971629958	-53376753	-420540093	985924908	-676874431	-40104889	632825246	452362286	-41888		
4732	624290441	-968748583	-910655572	891990040	-415752951	-398302243	-709366233	-328050895	2	
90258462	727929944	-63838459	779123608	-384598168	833283608	-840500486	368050357	158677		
079	-782579855	185049841	-667939146	79019831	-110742985	-730080913	985197687	-669474692	-	
720944587	591107795	-509301839	-305908586	858928446	-194709701	146256319	-136657888	-59762		
746	-311338998	-174618127	858878164	-99780380	-297032629	86890602	-29934025	-105051038	-	
422369120	-386484347	839627057	994010229	859846396	-948030026	757398982	-37709867	957159		
860	20977564	-128648736	-563036063	-742853120	230408535	579827128	759863796	2629888	-25073	
3905	216469800	405759120	-998149617	-839686821	-117300100	393723985	-905065700	-154523536	-	
672055365	-613237088	817203782	-631748597	-115797508	153464438	-543384976	-461177385	932855		
757	-483099781	69896031	136793185	216007901	959496077	-989638833	-70119092	-974929169	6	
51583016	-765821079	855765944	425123610	-166325047	-712613071	-307680154	167265935	457496		
750	528560597	-65186472	-568435798	-850652882	512459204	561974140	-387954907	45319385	-	
450651539	-547106274	697635649	-873345612	-221650081	837419895	289225295	9213530	909728627	6	
25376386	141767053	-699428128	-126305139	416052843	721521418	928030191	-46211582	805680		
703	954717901	-630479968	-932694120	-805773110	720775581	877076705	-978741973	649158642	7	
77821647	-842201090	919512900	886523014	-700314128	584312604	-35564337	-522903397	-53479		
2976	750494282	-524492252	545409608	350071262	-337326926	778849545	115771199	-280320578	3	
16103781	157691227	51095348	-48813394	-85751354	-761857620	585082792	-102782368	-94365		
6360	-835606232	-188336425	12709252	-881301094	-15309513	-89123867	-865634906	55980305	1	
5914454	747347826	-690268562	58663323	-956889456	54506442	-744702997	-414529913	-771554552	-	
565154814	-92115784	-533016280	-243974937	-43978723	655466076	316575319	33416556	-25818		
8921	-86241159	196466189	-4808661	-456107672	-139810723	-27890401	-461432115	27923724	-	
832642152	-468286583	-887476235	550719895	70591831	-223630549	-200498965	727374979	-22173		
7768	300829094									

Press ENTER to continue...

Рисунок 2 - Результат работы программы 2

3.3 Для программы, описанной в пункте 3.2, мы добавили определения времени создания и заполнения каждого из трех векторов с помощью библиотеки chrono. Также не забыли собрать программу в конфигурации «Release». Пример работы программы представлен на рисунке 3. Полный текст представлен в приложении В.

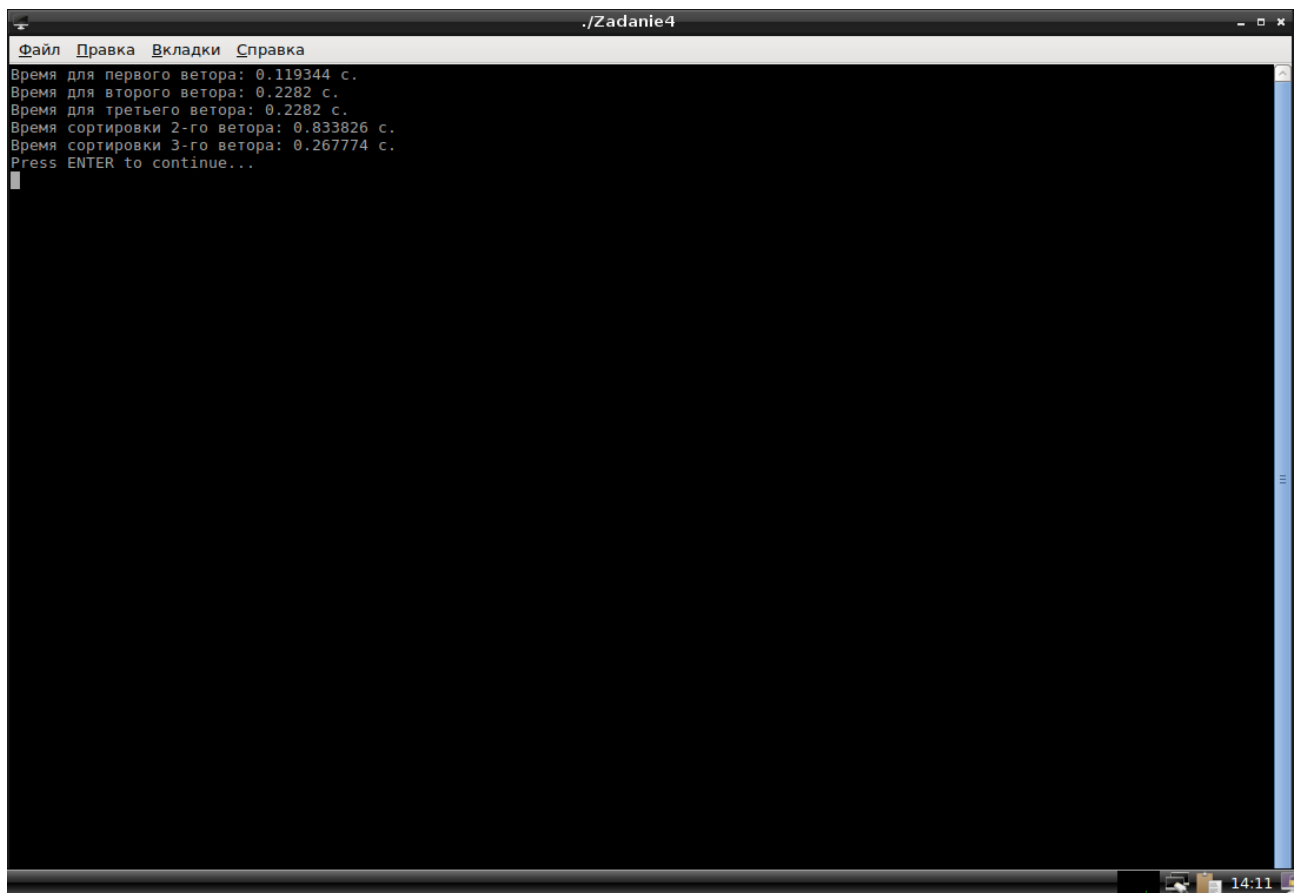


```
Файл  Правка  Вкладки  Справка
Время для первого вектора: 0.123181 с.
Время для второго вектора: 0.227952 с.
Время для третьего вектора: 0.227952 с.
Press ENTER to continue...
```

Рисунок 3 - Результат работы программы 3

При выполнении программы несколько раз, мы заметили, что 2 и 3 вектор заполняются всегда с одинаковым временем.

3.4 Для программы, описанной в пункте 3.3, мы добавили сортировку второго и третьего векторов, а также определили время каждой сортировки. Один вектор отсортировали с помощью алгоритма `sort`, второй с помощью алгоритма `stable_sort`. Пример работы программы представлен на рисунке 4. Полный текст представлен в приложении Г.

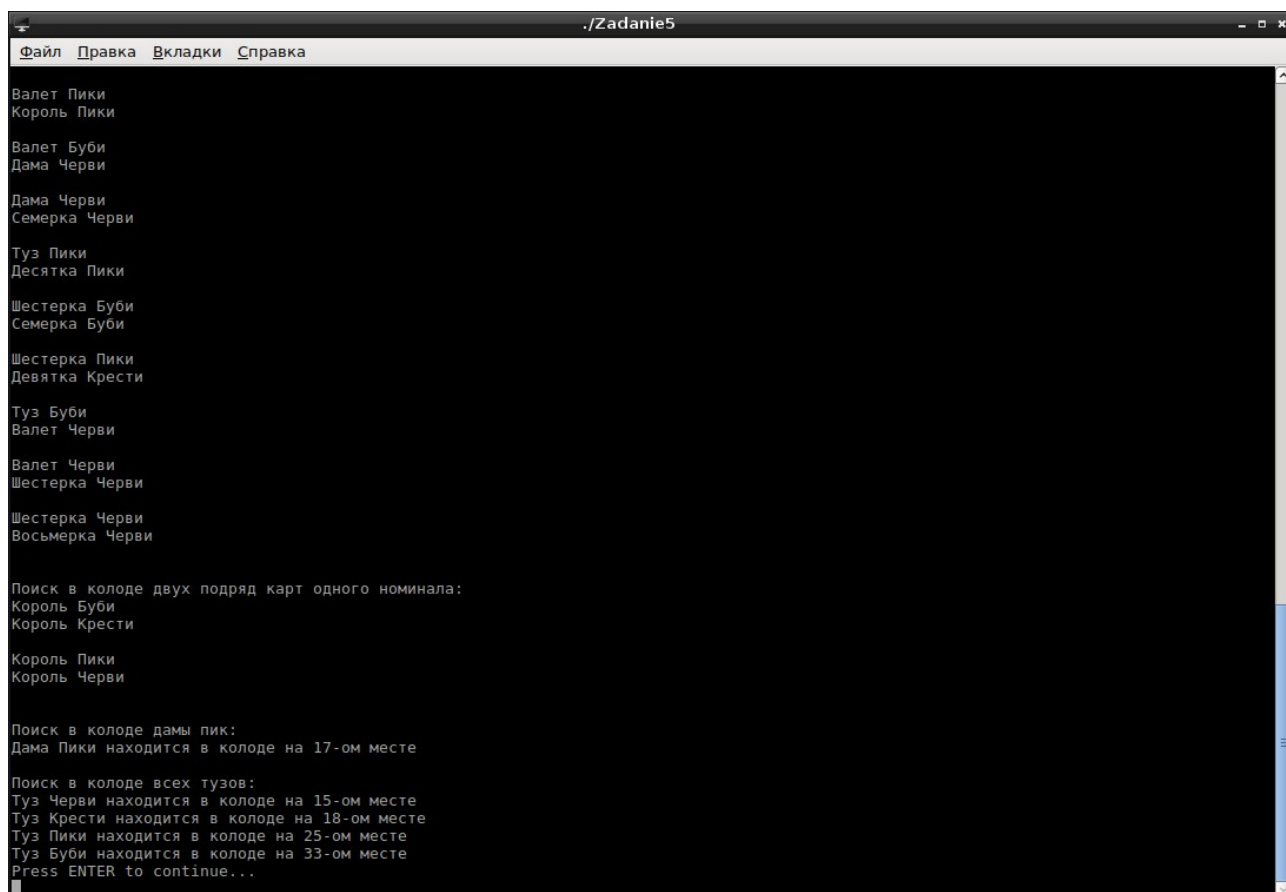


```
./Zadanie4
Файл  Правка  Вкладки  Справка
Время для первого ветора: 0.119344 с.
Время для второго ветора: 0.2282 с.
Время для третьего ветора: 0.2282 с.
Время сортировки 2-го ветора: 0.833826 с.
Время сортировки 3-го ветора: 0.267774 с.
Press ENTER to continue...
```

Рисунок 4 - Результат работы программы 4

Из результатов выполнения программы видно, что сортировка `stable_sort` затрачивает значительно меньше времени, чем обычная сортировка `sort`.

3.5 Для программы была выбрана колода из 36 карт. Была создана структура с двумя переменными типа `integer` `real_mast` и `real_number`, значение масти и номинала соответственно. Для перемешивания колоды использовался алгоритм `shuffle` с использованием генератора ПСП `mt19937`. Пример работы программы представлен на рисунке 5. Полный текст представлен в приложении Д.



```
./Zadanie5
Файл  Правка  Вкладки  Справка

Валет Пики
Король Пики

Валет Буби
Дама Черви

Дама Черви
Семерка Черви

Туз Пики
Десятка Пики

Шестерка Буби
Семерка Буби

Шестерка Пики
Девятка Крести

Туз Буби
Валет Черви

Валет Черви
Шестерка Черви

Шестерка Черви
Восьмерка Черви

Поиск в колоде двух подряд карт одного номинала:
Король Буби
Король Крести

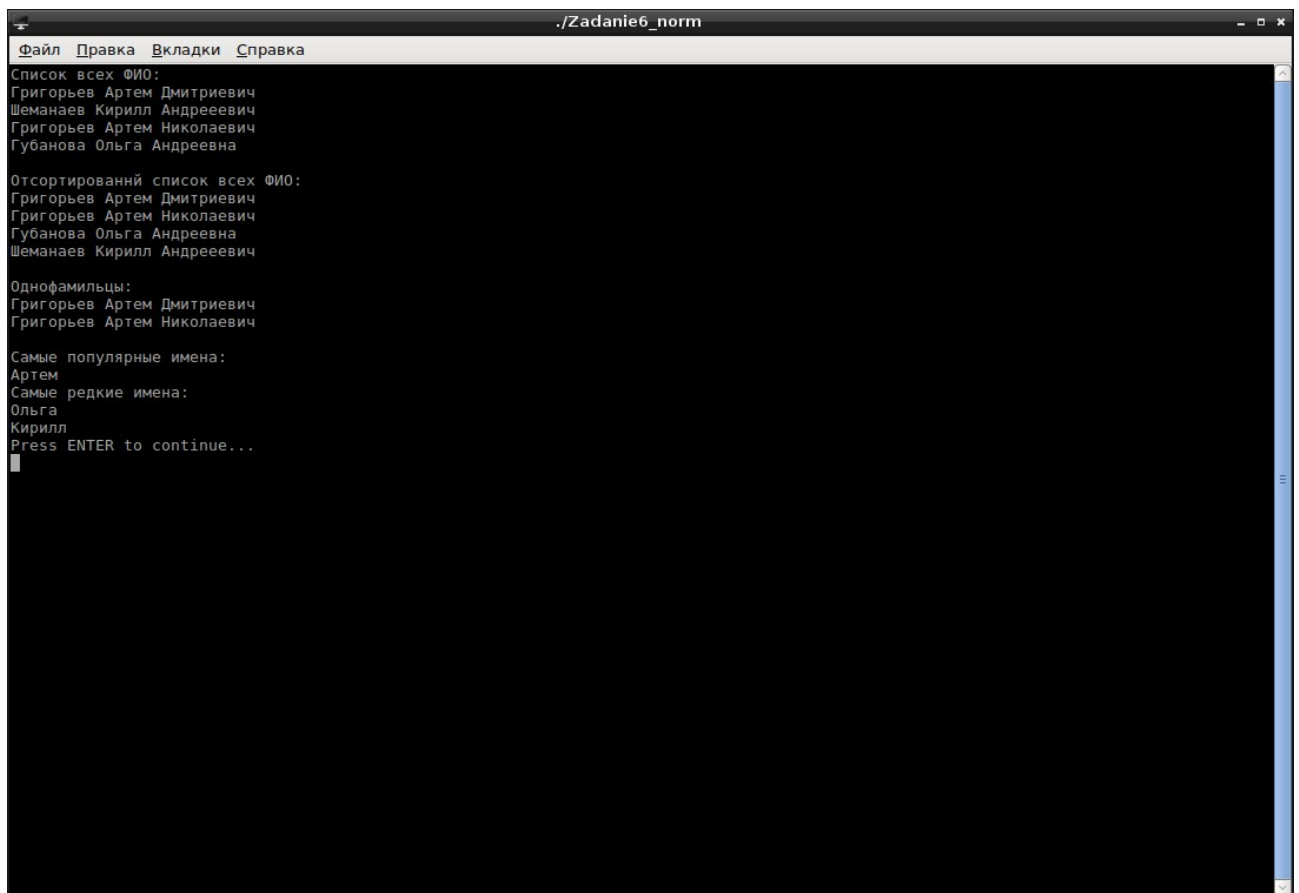
Король Пики
Король Черви

Поиск в колоде дамы пик:
Дама Пики находится в колоде на 17-ом месте

Поиск в колоде всех тузов:
Туз Черви находится в колоде на 15-ом месте
Туз Крести находится в колоде на 18-ом месте
Туз Пики находится в колоде на 25-ом месте
Туз Буби находится в колоде на 33-ом месте
Press ENTER to continue...
```

Рисунок 5 - Результат работы программы 5

3.6 В программе ФИО считываются из файла и заносятся в вектор с помощью метода `push_back()`. Для сортировки вектора по фамилиям использовалась сортировка пузырьком. Пример работы программы представлен на рисунке 6. Полный текст представлен в приложении Е.



```
./Zadanie6_norm
Файл  Правка  Вкладки  Справка
Список всех ФИО:
Григорьев Артем Дмитриевич
Шеманаев Кирилл Андреевич
Григорьев Артем Николаевич
Губанова Ольга Андреевна

Отсортированный список всех ФИО:
Григорьев Артем Дмитриевич
Григорьев Артем Николаевич
Губанова Ольга Андреевна
Шеманаев Кирилл Андреевич

Однофамильцы:
Григорьев Артем Дмитриевич
Григорьев Артем Николаевич

Самые популярные имена:
Артем
Самые редкие имена:
Ольга
Кирилл
Press ENTER to continue...
```

Рисунок 6 - Результат работы программы 6

4 Вывод

В результате выполнения работы было изучено использование стандартной библиотеки Си++, а также была написана программа для заполнения колоды карт, и получены практические навыки в использовании стандартных алгоритмов языка с++.

Приложение А.

Текст программы функции ctime на русском

```
#include <iostream>
#include <ctime>
using namespace std;

void get_time(tm * ptm);

int main(int argc, char **argv)
{

    time_t t = time(NULL);
    tm * ptm;
    ptm = localtime(&t);
    get_time(ptm);
    return 0;
}

void get_time(tm * ptm)
{
    string s;
    switch (ptm->tm_wday) {
    case 0:
        s += "Воскресенье";
        break;
    case 1:
        s += "Понедельник";
        break;
    case 2:
        s += "Вторник";
        break;
    case 3:
        s += "Среда";
```

```

        break;
case 4:
    s += "Четверг";
    break;
case 5:
    s += "Пятница";
    break;
case 6:
    s += "Суббота";
    break;
}

```

```

switch (ptm->tm_mon) {
case 0:
    s += " Январь";
    break;
case 1:
    s += " Февраль";
    break;
case 2:
    s += " Март";
    break;
case 3:
    s += " Апрель";
    break;
case 4:
    s += " Май";
    break;
case 5:
    s += " Июнь";
    break;
case 6:
    s += " Июль";

```

```

        break;
    case 7:
        s += " Август";
        break;
    case 8:
        s += " Сентябрь";
        break;
    case 9:
        s += " Октябрь";
        break;
    case 10:
        s += " Ноябрь";
        break;
    case 11:
        s += " Декабрь";
        break;
    }

    cout << s << " " << ptm->tm_mday << " " << ptm->tm_hour << ":"
<< ptm->tm_min << ":" << ptm->tm_sec << " " << ptm->tm_year+1900
<< endl;
}

```

Приложение Б.

Текст программы заполнения вектора случайными значениями

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <random>
using namespace std;

int RandomGenerator()
{
    static mt19937 rnd((uint64_t)&rnd);
    uniform_int_distribution<int> d(-1000000000, 1000000000);
    return d(rnd);
}

int main()
{
    vector<int> v;
    mt19937 rnd(-1000000000);
    for (int i=0; i < 10000000; i++)
        v.push_back(rnd());
    vector<int> v1(10000000);
    generate(v1.begin(), v1.end(), RandomGenerator);
    vector<int> v2(v1);
    for(auto e:v2)
        cout<<e<<'\t';
    cout<<endl;
}
```

Приложение В.

Текст программы для подсчета времени заполнения векторов

```
#include <algorithm>
#include <vector>
#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()
{
    random_device rd;
    vector<int> v;
    mt19937 rnd(-1000000000);
    steady_clock::time_point tp1 = steady_clock::now();
    for (int i=0; i < 10000000; i++)
        v.push_back(rnd());
    steady_clock::time_point tp2 = steady_clock::now();
    duration<double> d = tp2 - tp1;
    cout << "Время для первого ветора: " << d.count() << " с." <<
endl;
    vector<int> v1(10000000);
    tp1 = steady_clock::now();
    generate(v1.begin(), v1.end(), RandomGenerator);
    tp2 = steady_clock::now();
    d = tp2 - tp1;
```

```
        cout << "Время для второго ветора: " << d.count() << " с." <<
endl;
        tp1 = steady_clock::now();
        vector<int> v2(v1);
        tp2 = steady_clock::now();
        cout << "Время для третьего ветора: " << d.count() << " с." <<
endl;
    }
```

Приложение Г.

Текст программы для сортировки векторов

```
#include <iostream>
#include <algorithm>
#include <vector>
#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()
{
    random_device rd;
    vector<int> v;
    mt19937 rnd(-1000000000);
    steady_clock::time_point tp1 = steady_clock::now();
    for (int i=0; i < 10000000; i++)
        v.push_back(rnd());
    steady_clock::time_point tp2 = steady_clock::now();
    duration<double> d = tp2 - tp1;
    cout << "Время для первого ветопа: " << d.count() << " с." <<
endl;
    vector<int> v1(10000000);
    tp1 = steady_clock::now();
    generate(v1.begin(), v1.end(), RandomGenerator);
    tp2 = steady_clock::now();
```

```

    d = tp2 - tp1;
    cout << "Время для второго ветора: " << d.count() << " с." <<
endl;
    tp1 = steady_clock::now();
    vector<int> v2(v1);
    tp2 = steady_clock::now();
    cout << "Время для третьего ветора: " << d.count() << " с." <<
endl;
    tp1 = steady_clock::now();
    sort(v1.begin(),v1.end());
    tp2 = steady_clock::now();
    d = tp2 - tp1;
    cout << "Время сортировки 2-го ветора: " << d.count() << " с."
<< endl;
    tp1 = steady_clock::now();
    stable_sort(v1.begin(), v1.end());
    tp2 = steady_clock::now();
    d = tp2 - tp1;
    cout << "Время сортировки 3-го ветора: " << d.count() << " с."
<< endl;
}

```


Приложение Д.

Текст программы для колоды карт

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <random>

using namespace std;

string    number[9]    {"Шестерка",    "Семерка",    "Восьмерка",
"Девятка","Десятка", "Валет", "Дама", "Король", "Туз"};
string mast[4] {"Крести", "Черви", "Буби", "Пики"};

struct karta {
    int real_mast;
    int real_number;
};

void fill_koloda(vector<karta>& t);
void mix_koloda(vector<karta>& t);
void find_color(vector<karta>& t); //Для первого вхождения в колоде
void find_mast(vector<karta>& t); //Для первого вхождения в колоде
void find_dama(vector<karta>& t);
void print_ace(vector<karta>& t);
void print_koloda(vector<karta>& t);

int main()
{
    vector<karta> koloda36(36);
    cout << "Заполнение колоды:" << endl;
    fill_koloda(koloda36);
    print_koloda(koloda36);
}
```

```

    cout << endl;
    cout << "Перемешивание колоды:" << endl;
    mix_koloda(koloda36);
    print_koloda(koloda36);
    cout << endl;
    cout << "Поиск в колоде двух подряд карт одного цвета:" <<
endl;
    find_color(koloda36);
    cout << endl;
    cout << "Поиск в колоде двух подряд карт одного номинала:" <<
endl;
    find_mast(koloda36);
    cout << endl;
    cout << "Поиск в колоде дамы пик:" << endl;
    find_dama(koloda36);
    cout << endl;
    cout << "Поиск в колоде всех тузов:" << endl;
    print_ace(koloda36);
    return 0;
}

void fill_koloda(vector<karta>& t)
{
    int i;
    int mast1 = 0;
    int number1 = 6;
    for(i = 0; i < 36; i++) {
        t[i].real_number = number1;
        t[i].real_mast = mast1;
        if (mast1 == 3) {
            mast1 = 0;
            number1++;
        } else

```

```

        mast1++;
    }
}

void mix_koloda(vector<karta>& t)
{
    static mt19937 rnd((uint64_t)&rnd);
    shuffle(t.begin(), t.end(), rnd);
}

void find_color(vector<karta>& t)
{
    int i;
    for(i = 1; i < 36; i++) {
        if ((t[i].real_mast == 1 && t[i-1].real_mast == 2) ||
            (t[i].real_mast == 0 && t[i-1].real_mast == 3) ||
            (t[i].real_mast == 1 && t[i-1].real_mast == 1) ||
            (t[i].real_mast == 2 && t[i-1].real_mast == 2) ||
            (t[i].real_mast == 3 && t[i-1].real_mast == 3) ||
            (t[i].real_mast == 4 && t[i-1].real_mast == 4))
        {
            cout << number[t[i - 1].real_number - 6] << " " <<
            mast[t[i - 1].real_mast] << endl;
            cout << number[t[i].real_number - 6] << " " <<
            mast[t[i].real_mast] << endl;
            cout << endl;
        }
    }
}

void find_mast(vector<karta>& t)
{
    int i;

```

```

        for(i = 1; i < 36; i++) {
            if (t[i].real_number == t[i-1].real_number) {
                cout << number[t[i] - 1].real_number - 6] << " " <<
mast[t[i] - 1].real_mast] << endl;
                cout << number[t[i].real_number - 6] << " " <<
mast[t[i].real_mast] << endl;
                cout << endl;
            }
        }
    }

void find_dama(vector<karta>& t)
{
    int i;
    for(i = 0; i < 36; i++) {
        if (t[i].real_number == 12 && t[i].real_mast == 3) {
            cout << number[t[i].real_number - 6] << " " <<
mast[t[i].real_mast] << " находится в колоде на " << i + 1 << "-ом
месте" << endl;
            break;
        }
    }
}

void print_ace(vector<karta>& t)
{
    int i;
    for(i = 0; i < 36; i++) {
        if (t[i].real_number == 14 && t[i].real_mast == 0) {
            cout << number[t[i].real_number - 6] << " " <<
mast[t[i].real_mast] << " находится в колоде на " << i + 1 << "-ом
месте" << endl;
        }
    }
}

```

```

        if (t[i].real_number == 14 && t[i].real_mast == 1) {
            cout << number[t[i].real_number - 6] << " " <<
mast[t[i].real_mast] << " находится в колоде на " << i + 1 << "-ом
месте" << endl;
        }
        if (t[i].real_number == 14 && t[i].real_mast == 2) {
            cout << number[t[i].real_number - 6] << " " <<
mast[t[i].real_mast] << " находится в колоде на " << i + 1 << "-ом
месте" << endl;
        }
        if (t[i].real_number == 14 && t[i].real_mast == 3) {
            cout << number[t[i].real_number - 6] << " " <<
mast[t[i].real_mast] << " находится в колоде на " << i + 1 << "-ом
месте" << endl;
        }
    }
}

void print_koloda(vector<karta>& t)
{
    int i;
    for(i = 0; i < 36; i++) {
        cout << number[t[i].real_number - 6] << " " <<
mast[t[i].real_mast] << endl;;
    }
}

```

Приложение Е.

Текст программы для считывания из файла списка людей

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <vector>
using namespace std;

void bubblesort(vector<string>& mass, unsigned size);
void find_surname(vector<string>& mass, unsigned size);
void find_name(vector<string>& mass, int size);
int space(string s, unsigned begin = 0);

int main(int argc, char **argv)
{
    unsigned i = 0;
    ifstream f("/root/Laba9/Zadanie6/data_v1");
    vector<string> s;
    string line;
    while (getline(f, line)) {
        if (line != "" || line != " " || line != "\n")
            s.push_back(line);
    }
    cout << "Список всех ФИО:" << endl;
    for (i = 0; i < s.size(); i++)
        cout << s[i] << endl;
    cout << endl;
    bubblesort(s, s.size());
    cout << "Отсортированный список всех ФИО:" << endl;
    for (i = 0; i < s.size(); i++)
        cout << s[i] << endl;
    cout << endl;
```

```

        cout << "Однофамильцы:" << endl;
        find_surname(s, s.size());
        cout << endl;
        find_name(s, s.size());
        f.close();
        return 0;
    }

void bubblesort(vector<string>& mass, unsigned size)
{
    unsigned i,j;
    bool swapped;
    for(i = 0; i < size-1; i++) {
        swapped = false;
        for(j = 0; j < size-i-1; j++) {
            if (mass[j] > mass[j + 1]) {
                string Temp = mass[j];
                mass[j] = mass[j + 1];
                mass[j + 1] = Temp;
                swapped = true;
            }
        }
        if (swapped == false)
            break;
    }
}

void find_surname(vector<string>& mass, unsigned size)
{
    string surname1, surname2;
    int probel;
    unsigned i = 0, j = 0;
    for(i = 0; i < size; i++) {

```

```

        probel = space(mass[i]);
        surname1 = mass[i].substr(0,probel);
        for(j = i + 1; j < size; j++) {
            probel = space(mass[j]);
            surname2 = mass[j].substr(0,probel);
            if (surname1 == surname2) {
                cout << mass[i] << endl;
                cout << mass[j] << endl;
            }
        }
    }
}

int space(string s, unsigned begin)
{
    int i = 0;
    if (begin > 0) {
        begin++;
        i++;
    }
    while (s[begin] != ' ') {
        i++;
        begin++;
    }
    return i;
}

void find_name(vector<string>& mass, int size)
{
    string name1, name2;
    int probel1, probel2;
    int i = 0, j = 0, max = -1, min = size + 1;
    int name_count[size];

```



```

for (i = 0; i < size; i++)
    name_count[i] = 0;
for (i = 0; i < size; i++) {
    probell1 = space(mass[i]);
    probel2 = space(mass[i], probell1);
    name1 = mass[i].substr(probell1, probel2);
    for (j = 0; j < size; j++) {
        probell1 = space(mass[j]);
        probel2 = space(mass[j], probell1);
        name2 = mass[j].substr(probell1, probel2);
        if ((mass[i] != mass[j]) && (name1 == name2))
            name_count[i]++;
    }
}

for (i = 0; i < size; i++) {
    if (name_count[i] > max)
        max = name_count[i];
    else if (name_count[i] < min)
        min = name_count[i];
}

//max
name1 = "";
name2 = "";
cout << "Самые популярные имена:" << endl;
for (i = 0; i < size; i++) {
    if (name_count[i] == max) {
        probell1 = space(mass[i]);
        probel2 = space(mass[i], probell1);
        name1 = mass[i].substr(probell1 + 1, probel2);
        if (name1 == name2)
            continue;
        cout << name1 << endl;
        name2 = name1;
    }
}

```

```

        }
    }
    //min
    name1 = "";
    name2 = "";
    cout << "Самые редкие имена:" << endl;
    for (i = 0; i < size; i++) {
        if (name_count[i] == min) {
            probel1 = space(mass[i]);
            probel2 = space(mass[i], probel1);
            name1 = mass[i].substr(probel1 + 1, probel2);
            if (name1 == name2)
                continue;
            cout << name1 << endl;
            name2 = name1;
        }
    }
}

```