

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

ОТЧЕТ  
о практическом задании  
ИСПОЛЬЗОВАНИЕ СТАНДАРТНОЙ БИБЛИОТЕКИ C++

Дисциплина: Языки программирования

Группа: 18ПИ1

Выполнил: Водянов В.О.

Количество баллов:

Дата сдачи:

Принял: к.т.н., доцент М.Ю. Лупанов

Пенза, 2019г

## 1. Цель работы

Знакомство с возможностями и использование стандартной библиотеки C++.

## 2. Задания к практической 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. Результат выполнения работы

3.1. Написана программа для вывода текущего времени, названий дней недели и месяцев на русском языке. Использована функция `strftime()` и спецификатор `%c`. Для определения времени применена функция `localtime()`.

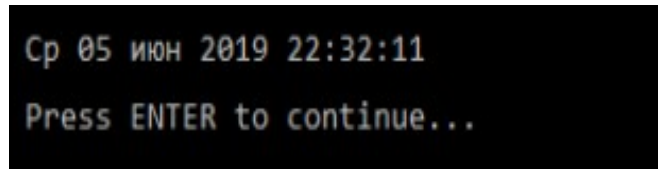


Рисунок 1 – Результат работы программы для вывода времени и даты

Текст программы:

```
#include <ctime>
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
    locale loc("ru_RU.UTF-8");
    locale::global(loc);
    time_t t = time(NULL);
    tm* ptm;
    ptm = localtime(&t);
    char buf[80];
    strftime(buf, 80, "%c", ptm);
    cout << buf << endl;
    return 0;
}
```

3.2. Написана программа, формирующая три целочисленных вектора. Первый вектор пустой. Генерация значений осуществима при помощи `mt19937` и равномерного распределения целых чисел. Второй вектор заполнен сразу при помощи алгоритма `generate` и функции `RandomGenerator()`. Третий же вектор был получен с помощью конструктора создания вектора-копии.

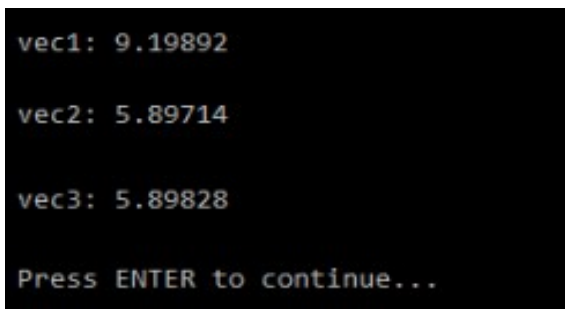
Текст программы:

```

#include <algorithm>
#include <iostream>
#include <random>
#include <vector>
using namespace std;
int RandomGenerator()
{
    static mt19937 rnd(50);
    uniform_int_distribution<int> r(-1000000000, 1000000000);
    return r(rnd);
}
int main(int argc, char** argv)
{
    vector<int> vec1;
    vector<int> vec2(10000000);
    mt19937 rnd(123);
    uniform_int_distribution<int> roll(-1000000000, 1000000000);
    for(int i = 0; i < 10000000; i++)
        vec1.push_back(roll(rnd));
    generate(vec2.begin(), vec2.end(), RandomGenerator);
    vector<int> vec3(vec2);
    return 0;
}

```

3.3. Написана программа уточняющая программу из пункта 2.2. Для определения времени перед каждым заполнением вектора и после, поставлены временные точки, изменено время между ними. Результат работы программы представлен на рисунке 2.



```

vec1: 9.19892
vec2: 5.89714
vec3: 5.89828
Press ENTER to continue...

```

Рисунок 2 — Результат работы программы

Наиболее медленное заполнение у первого вектора, так как затрачивается время на постоянное расширение памяти. Для заполнения второго вектора память была выделена при инициализации, поэтому он заполняется быстрее.

Текст программы:

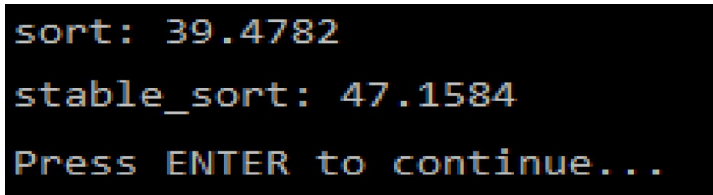
```
#include <algorithm>
#include <chrono>
#include <iostream>
#include <random>
#include <vector>
using namespace std;
using namespace std::chrono;
int RandomGenerator()
{
    static mt19937 rnd(122); //(uint64_t)&rnd);
    uniform_int_distribution<int> r(-1000000000, 1000000000);
    return r(rnd);
}
int main(int argc, char** argv)
{
    vector<int> vec1;
    mt19937 rnd(50);
    uniform_int_distribution<int> roll(-1000000000, 1000000000);
    steady_clock::time_point t_p1 = steady_clock::now();
    for(int i = 0; i < 10000000; i++)
        vec1.push_back(roll(rnd));
    steady_clock::time_point t_p2 = steady_clock::now();
    duration<double> r = t_p2 - t_p1;
    cout << "vec1: " << r.count() << endl;
    t_p1 = steady_clock::now();
    vector<int> vec2(10000000);
    generate(vec2.begin(), vec2.end(), RandomGenerator);
    t_p2 = steady_clock::now();
    r = t_p2 - t_p1;
    cout << "vec2: " << r.count() << endl;
    t_p2 = steady_clock::now();
    vector<int> vec3(vec2);
```

```

    r = t_p2 - t_p1;
    cout << "vec3: " << r.count() << endl;
    return 0;
}

```

3.4. В программу добавлены две сортировки векторов `sort` `stable_sort`. Измерения проводятся также как и в предыдущем пункте. Результат работы программы представлен на рисунке 3.



```

sort: 39.4782
stable_sort: 47.1584
Press ENTER to continue...

```

Рисунок 3 — Результат работы программы

Для того, чтобы использовать сортировку типа `stable_sort` нужно доп. пространство, поэтому она медленнее, чем `sort`.

Текст программы:

```

#include <algorithm>
#include <chrono>
#include <iostream>
#include <random>
#include <vector>
using namespace std;
using namespace std::chrono;
int RandomGenerator()
{
    static mt19937 rnd(50); //(uint64_t)&rnd);
    uniform_int_distribution<int> r(-1000000000, 1000000000);
    return r(rnd);
}
int main(int argc, char** argv)
{
    vector<int> vec1;
    vector<int> vec2(10000000);
    mt19937 rnd(50);
    uniform_int_distribution<int> roll(-1000000000, 1000000000);

```

```

for(int i = 0; i < 10000000; i++)
    vec1.push_back(roll(rnd));
generate(vec2.begin(), vec2.end(), RandomGenerator);
vector<int> vec3(vec2);
steady_clock::time_point t_p1 = steady_clock::now();
sort(vec2.begin(), vec2.end());
steady_clock::time_point t_p2 = steady_clock::now();
duration<double> r = t_p2 - t_p1;
cout << "sort: " << r.count() << endl;
t_p1 = steady_clock::now();
stable_sort(vec3.begin(), vec3.end());
t_p2 = steady_clock::now();
r = t_p2 - t_p1;
cout << "stable_sort: " << r.count() << endl;
return 0;
}

```

3.5. Реализована структура Card. В ней хранятся атрибуты масти и ранги карт. Создан вектор этих структур. Добавлен перегруженный оператор вывода. С помощью реализации алгоритма random\_shuffle реализовано перемешивание колоды. Алгоритм adjacent\_find и функция Color реализованы для поиска карт одинакового цвета. С помощью алгоритма adjacent\_find и функции Nominal идет поиск карт одинакового номинала. Для поиска пиковой дамы используется алгоритм find\_if и функция Queen. Поиск тузов идет в цикле до тех пор пока не будут найдены все тузы. Результат работы программы представлен на рисунке 4.

```

7 1
Карты одного цвета: 6 1 и 2 1
Карты одного цвета: 8 3 и 6 2
Карты одного цвета: 4 3 и 8 2
Карты одного цвета: 6 3 и 4 3
Карты одного цвета: 1 2 и 4 2
Карты одного цвета: 0 0 и 2 0
Карты одного цвета: 7 2 и 5 2
Карты одного цвета: 0 2 и 7 2
Карты одного цвета: 5 3 и 0 2
Карты одного цвета: 5 0 и 8 0
Карты одного цвета: 3 1 и 7 0
Карты одного цвета: 5 1 и 3 1
Карты одного цвета: 0 3 и 2 2
Карты одного цвета: 1 1 и 6 0
Карты одного цвета: 4 1 и 0 1
Карты одного цвета: 7 1 и 3 0
Карты одного номинала: 6 2 и 6 1
Карты одного номинала: 8 1 и 8 3
Карты одного номинала: 1 0 и 1 1
Пиковая дама 10-я в колоде
Тузы в колоде на 5 6 8 19 позициях
Press ENTER to continue...

```

Рисунок 4 — Результат работы программы

Текст программы:

```

#include <algorithm>
#include <chrono>
#include <iostream>
#include <random>
#include <vector>
using namespace std;
using namespace std::chrono;
int RandomGenerator()
{
    static mt19937 rnd(50); //(uint64_t)&rnd);
    uniform_int_distribution<int> r(-1000000000, 1000000000);
    return r(rnd);
}
int main(int argc, char** argv)
{
    vector<int> vec1;
    vector<int> vec2(10000000);
    mt19937 rnd(50);
    uniform_int_distribution<int> roll(-1000000000, 1000000000);
    for(int i = 0; i < 10000000; i++)
        vec1.push_back(roll(rnd));
}

```



```

generate(vec2.begin(), vec2.end(), RandomGenerator);
vector<int> vec3(vec2);
steady_clock::time_point t_p1 = steady_clock::now();
sort(vec2.begin(), vec2.end());
steady_clock::time_point t_p2 = steady_clock::now();
duration<double> r = t_p2 - t_p1;
cout << "sort: " << r.count() << endl;
t_p1 = steady_clock::now();
stable_sort(vec3.begin(), vec3.end());
t_p2 = steady_clock::now();
r = t_p2 - t_p1;
cout << "stable_sort: " << r.count() << endl;
return 0;
}

```

3.6. Написана программа, читающая из файла ФИО и выполняющая определенные действия. Для хранения списка людей создан контейнер `set` типа `wstring`. С помощью методов `getline` и `insert` в контейнер имена людей были записаны до `'\n'`. Изначально выполнена сортировка по фамилии, так как используется `set`. С помощью алгоритма `adjacent_find` и функции `Familya` выполнен поиск однофамильцев. Создан контейнер `map` для хранения имен. Поэтому при повторении ключа, второе значение принимает свое значение на единицу больше. С помощью алгоритмов `max_element` и `min_element` и функций `pName` и `rName` осуществлен поиск наиболее часто и наиболее редко упоминающихся имен.

Текст программы:

```

#include <algorithm>
#include <fstream>
#include <iostream>
#include <map>
#include <set>
#include <string>
using namespace std;
bool Familya(const wstring& wstr1, const wstring& wstr2)
{
    int pos1 = wstr1.find(L" ");

```

```

        int pos2 = wstr2.find(L" ");
        return (wstr1.substr(0, pos1) == wstr2.substr(0, pos2));
    }
    bool rName(const pair<wstring, int>& pr1, const pair<wstring,
int>& pr2)
    {
        return pr1.second < pr2.second;
    }
    bool pName(const pair<wstring, int>& pr1, const pair<wstring,
int>& pr2)
    {
        return pr1.second < pr2.second;
    }
    wstring Name(wstring& t)
    {
        int start_pos = t.find(L" ") + 1;
        int pos = t.find(L" ", start_pos) - start_pos;
        return t.substr(start_pos, pos);
    }
    int main(int argc, char** argv)
    {
        locale loc("ru_RU.UTF-8");
        locale::global(loc);
        set<wstring> pr;
        wifstream d("/home/user/doc.txt");
        while(!d.eof()) {
            wstring t;
            getline(d, t);
            pr.insert(t);
        }
        for(auto it = ++pr.begin(); it != pr.end(); it + +) {
            it = adjacent_find(--it, pr.end(), Familya);
            wcout << L"Однофамильцы: " << *(it) << L" и " << *(it++) <<
endl;
        }
        map<wstring, int> name;
        map<wstring, int>::iterator it;

```

```

for(auto i : pr) {
    name[Name(i)]++;
}
auto it1 = max_element(name.begin(), name.end(), pName);
auto it2 = min_element(name.begin(), name.end(), rName);
wcout << L"Часто: " << endl;
for(it = name.begin(); it != name.end(); it++) {
    if(it->second == it1->second)
        wcout << it->first << endl;
}
wcout << L"Редко: " << endl;
for(it = name.begin(); it != name.end(); it++) {
    if(it->second == it2->second)
        wcout << it->first << endl;
}
d.close();
return 0;
}

```

#### 4. Вывод

Были изучены возможности стандартной библиотеки C++. Освоены алгоритмы. Получены практические навыки по использованию стандартной библиотеки C++.