Контрольная по STL 2017(на 9-10)

без различия регистров

/*Часть 1
Разработать класс для хранения данных. Заполнить последовательный контейнер
записями из файла DATA.TXT.
Сортировка.

В файле размещаются данные об автобусных маршрутах в формате (построчно):
№маршрута;название;перечень водителей, работающих на маршруте
Данные разделены точкой с запятой (в том числе и имена водителей),
все остальные символы являются частью данных.
В конце точки с запятой нет. Все строки различны, но неотсортированы.

- Разместить данные в list. Для хранения имен водителей использовать vector.
- Выполнить сортировку по названию маршрута (в алфавитном порядке),
для одинаковых названий – по возрастанию №маршрута.

Часть 2
Использование ассоциативных контейнеров. Данные для поиска задавать в программе.

- Вывести список водителей без повторов.
- Найти маршруты, на которых не работает заданный водитель.
- Найти водителей, которые могут работать на всех маршрутах.

Часть 3
Использование стандартных алгоритмов. Данные для поиска задавать в программе.

- Подсчет числа маршрутов по заданному названию.
- Удаление указанного водителя.

```
Вывод результатов в файл ANSWER.TXT в форме, удобной
для чтения.
Предусмотреть отсутствие информации.
При выводе разделять выводимые результаты. Например,
Результаты части №1
……..
Результаты части №2
……
Результаты части №3

На 9-10 – прописные и строчные буквы в именах и
названиях не различать! Кириллица.
При выводе сохранять написание из исходного
файла.
Естественно, проверка на полноту тестов (все возможные
случаи д.б. учтены)*/

#include <iostream>
#include <fstream>
#include <map>
#include <vector>
#include <set>
#include <list>
#include <algorithm>
#include <string>
#include <utility>
#include <functional>
#include <iterator>
#include <stdlib.h>

using namespace std;

class Bas;
class LessString;
class Compare_Register;
typedef vector <string> V_Vodit;
typedef list <Bas> L_Bas;
typedef set <string,LessString> S_Set;

L_Bas L;
L_Bas::iterator it1;
V_Vodit Vect,Vect1,vect;
V_Vodit::iterator it3;
```

```
ofstream fout("ANSWER.TXT");
ifstream f;

class Bas
{
private:
     int M;
     string Nazv;
     V_Vodit Vect;
public:
     Bas():M(0),Nazv("noname"){}
     Bas(int m, string n, V_Vodit v)
     {
          M=m;
          Nazv=n;
          Vect=v;
     }
     V_Vodit getVect(){return Vect;};
     void setVect(V_Vodit v){Vect=v;};
     int getM() const {return M;};
     string getNazv() const {return Nazv;};
     friend bool operator < (const Bas&,const Bas&);
     void print() const;
};




/*****************************************/
void Bas::print()const
{
     cout << "маршрут номер " << M << " название " <<
               Nazv << "\nводители: ";
     copy(Vect.begin(), Vect.end(),
               ostream_iterator<string>(cout," "));
     cout << endl << endl;
}
```

```
/*****************************************************/
string My_Translate(string s)  // На 9-10 баллов
//перекодировка .866 в .1251 (Dos -> Windows)
//char со знаком, поэтому русские буквы имеют //
отрицательный код
{
     string x;
     x=s;
     for(unsigned i=0; i<x.length();i++)
          if ((x[i]>=-128)&&(x[i]<=-81))
               x[i]+=64;
          else
               if ((x[i]>=-32)&&(x[i]<=-17))
                    x[i]+=16;
     return x;
}

/*****************************************************/
static bool mycomp (const char c1, const char c2)
     { return tolower(c1)<tolower(c2); }

/*****************************************************/
static bool Ravno(const string s1, const string s2)
{
          if (s1.length()==s2.length())  // if (s1==s2)
               // используем алгоритм сравнения
               // по лексикографическому критерию.
          if (!lexicographical_compare (s1.begin(),
               s1.end(), s2.begin(), s2.end(), mycomp) &&
               !lexicographical_compare (s2.begin(),
                s2.end(),  s1.begin(), s1.end(), mycomp))
          return true;
          return false;
}

/*****************************************/
bool operator < (const Bas& x1, const Bas& x2)
 // для сортировки по названию маршрута и его номеру
{
     if(Ravno(x1.Nazv,x2.Nazv))
          return(x1.M < x2.M);
     return (lexicographical_compare (x1.Nazv.begin(),
               x1.Nazv.end(), x2.Nazv.begin(),
               x2.Nazv.end(), mycomp));
}
```

```
/*****************************************************/

class LessString: binary_function <string,  string,
bool>
                              //предикат для множества
{
public:
     bool operator()(string s1,  string s2)const
     {
          return lexicographical_compare (s1.begin(),
                 s1.end(), s2.begin(), s2.end(), mycomp);
     }
     friend static bool mycomp (const char c1,
                                const char c2);
};



/*****************************************************/

class Compare_Register  //функциональный объект
     // для поиска заданной строки без различия регистра
{
public:
     string Stroka;
     friend static bool Ravno(const string s1,
                              const string s2);
     friend static bool mycomp (const char c1,
                                const char c2);

     Compare_Register(string x)
     {
          Stroka=x;
     }
     bool operator()(Bas& z)
     {
          string ss=z.getNazv();
          return (Ravno(ss,Stroka));
     }
};
```

```
/*****************************************************/

int Input_file(L_Bas& L,char* fname)      // Для Части 1
//ввод из файла и формирование списка
{
     string marsh, nazvanie, voditel;
     V_Vodit vect;
     string s;
     int  imarsh;
     size_t i1, i2;
     f.open(fname,ios::in);
     if(!f)
     {
          cout<<"нет входного файла"<<endl;
          return -1;
     }

                              //чтение данных
     getline(f, s);
     if (!f)
     {
          cout << "файл пустой" << endl;
          return -1;
     }
     while (f)
     {
          i1=0;
          i2=s.find(';',0);
          marsh=s.substr(0,i2);
          i1=i2+1;
          i2=s.find(';',i1);
          nazvanie=s.substr(i1,i2-i1);
          i1=i2+1;
          vect.clear();
          do
          {
               i2=s.find(';',i1);
               if (i2!= -1) //не последний водитель
               {
                    voditel=s.substr(i1,i2-i1);
                    i1=i2+1;
               }
               else          // последний водитель
                    voditel=s.substr(i1,s.length()-i1);
```

```
                vect.push_back(voditel);
            }
            while(i2 != -1);
            imarsh=atoi(marsh.c_str());
                //!!!!!!!!!!!!!!!!!!!!!!!!!!
                    //cout<<imarsh<<endl;
            Bas bas(imarsh, nazvanie, vect);
                    //bas.print();
            L.push_back(bas);
            getline(f, s);
        }
        f.close();
        return 0;
}
```

```
/*****************************************/
```

```
void Poisk_all(L_Bas& L)     // Для Части 2
{    // поиск водителей, работающих на всех маршрутах
     V_Vodit Vect1;
     L_Bas::iterator it1 = L.begin();
     S_Set  Set1, Set2, SetA;
     Vect1 = (*it1).getVect();
     V_Vodit::iterator it3;
     for (unsigned ii = 0; ii<Vect1.size(); ii++)
                                          // по индексу
          SetA.insert(Vect1[ii]);
     cout << "Список  водителей, которые работают на
всех
               маршрутах:\n";
     for (it1=L.begin(), it1++; it1!=L.end(); it1++)
     {
          Set1.clear();
          Vect1=(*it1).getVect();
          for(it3 = Vect1.begin(), it3++; it3 !=
Vect1.end();
                    it3++) //по итератору
               Set1.insert(*it3);
          set_intersection(SetA.begin(),SetA.end(),
                           Set1.begin(),Set1.end(),
             inserter(Set2,Set2.begin()), LessString());
          SetA=Set2;
          Set2.clear();
     }
     if (SetA.size()==0)
          cout<<"нет таких водителей\n";
     else
     {
       copy(SetA.begin(), SetA.end(),
                    ostream_iterator<string> (cout," "));
       cout << endl;
     }
     cout << endl;
}
```

```
/*****************************************************/
void All_vodit(L_Bas& L, S_Set& Set)      // Для Части 2
{    // поиск всех водителей без повторений
     S_Set:: iterator it2;
     for (it1=L.begin(); it1!=L.end(); it1++)
     {
          Vect1=(*it1).getVect();
          for(unsigned ii=0; ii<Vect1.size();ii++)
               Set.insert(Vect1[ii]);
               // формирование множества всех водителей
     }
     if (Set.size()==0)
          cout<<"нет водителей вообще";
     else
     {
       cout << "Список всех водителей:\n";
       copy(Set.begin(), Set.end(),
            ostream_iterator<string> (cout," "));
     }
     cout << endl;
}
/*****************************************************/
void None(L_Bas& L, string fio)           // Для Части 2
{ // поиск маршрутов, на которых не работает заданный
  // водитель
     int kol=0;
     cout << endl;
     cout<<fio<<" не работает на следующих маршрутах:";
     for (it1=L.begin(); it1!=L.end(); it1++)
     {    Vect1=(*it1).getVect();
          bool pr=true;
          for(unsigned ii=0; ii<Vect1.size();ii++)
          {    string ss=Vect1[ii];
               // поиск заданного водителя
               if (Ravno(ss,fio))     //if
(Vect1[ii]==fio)
                    pr=false;
          }
          if (pr)
          {    cout<<(*it1).getM()<<' ';
               kol++;
          }
```

```
    }
    if (kol==0)  cout<<"нет таких маршрутов\n";
    cout << endl;
}
/*****************************************************/

void Count_Marsh(string nazvanie, L_Bas& L)     // Для
Части 3
//подсчет числа маршрутов по заданному названию
{
  int    count_marsh=count_if(L.begin(), L.end(),
         Compare_Register(nazvanie));
  if (count_marsh!=0)
    cout << "Количество маршрутов в " << nazvanie
         << " = " << count_marsh << endl;
  else
    cout << "В " << nazvanie << " нет маршрутов " <<
endl;
}
/*****************************************************/
void Erase_Vodit(string fio, L_Bas& L)    // Для Части 3
{   //удаление заданного водителя
    int kol=0;
    for (it1=L.begin(); it1!=L.end(); it1++)
    {
        Vect1=(*it1).getVect();
        //копир вектор в рабочий Vect1 и работаем с ним
        bool pr=false;
        for(it3=Vect1.begin(); it3!=Vect1.end(); it3++)
        {
            string ss=(*it3);
            if (Ravno(ss,fio))
            {
                pr=true;
                kol++;
                Vect1.erase(it3);
                break;
                //ибо водитель в векторе один раз встречается
            }
        }
        if (pr) //удаление из вектора  было
            (*it1).setVect(Vect1);
                //сохраняем измененный вектор
    }
```

```
     if (kol==0)
          cout<<"Нет водителя "  << fio <<
                " и список не изменился\n" << endl;
     else
       {
          cout << "Список после удаления водителя "
              << fio << ": " << endl;
          for_each(L.begin(),L.end(),
                   mem_fun_ref(&Bas::print));
             // вывод, используя for_each
             // используется функциональный адаптер
             // mem_fun_ref для
             // вызова Print как функции объекта
       }
}
/*****************************************************/

int main()
{
     setlocale(LC_ALL,".1251");
     if(!fout)
     {
          cout<<"нельзя создать выходной файл"<<endl;
          return -1;
     }
     cout<<"начало работы\nдальше вывод пойдет в
файл\n";
                                                  //!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

//freopen("ANSWER.TXT", "w", stdout);
//дальше вывод пойдет в файл
// перенаправление потока вывода с консоли в файл
// этот способ хорош, если до конца работы не будете
// выводить на консоль
// иначе будут проблемы с кирилицей, если вернуть вывод
// на консоль
// через freopen("CON","w",stdout);
// streambuf* our_buffer =cout.rdbuf(fout.rdbuf());
// вывод в файл в случае, если опять
//понадобится вывод на консоль кирилицы
```

```
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


//Часть 1
    cout << "Часть 1\n";
    L_Bas L;
    L_Bas::iterator it1;
    int rez=Input_file(L,"DATA.TXT");
                 // формирование списка
    if (rez==-1) return -1;
    cout<<"Список до сортировки:\n";
    for (it1=L.begin(); it1!=L.end(); it1++)
                 // вывод списка через цикл
        (*it1).print();
    L.sort();
    cout<<"Список после сортировки:\n";

for_each(L.begin(),L.end(),mem_fun_ref(&Bas::print));
                 // вывод, используя for_each

//Часть 2
    cout << "Часть 2\n";
    S_Set Set;  //множество всех водителей будет
    S_Set:: iterator it2;
// Поиск и вывод всех водителей
    All_vodit(L, Set);
//  int kol;
// на каких маршрутах не работает заданный водитель
    None(L, "ПЕТРОВ");
    None(L, "Зорин");
    None(L, "Жук");
// поиск водителей, работающих на всех маршрутах
// случай, когда есть такой
    Poisk_all( L);

//Часть 3
    cout << "Часть 3\n";
//подсчет числа маршрутов по заданному названию
    Count_Marsh("Михновичи",L); //есть такие
//подсчет числа маршрутов по заданному названию
    Count_Marsh("Михалово",L);  //нет таких

//удаление заданного водителя
    cout << "\nУдаление водителей\n";
```

```
    Erase_Vodit("Жук", L);       //есть такой водитель
    Erase_Vodit("Клюев", L);     //нет такого водителя
    Erase_Vodit("Давид", L);     //есть такой водитель

// часть 2 дополнение
// поиск водителей, работающих на всех маршрутах
// случай, когда нет такого (удалили его)
    cout << "К части 2 дополнение после удаления\n";
    Poisk_all( L);

//Часть 4
//проверка ввода с клавиатуры кирилицы
    string fio2, fio1; //
    //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    cout.rdbuf(our_buffer);
//перенаправление потока обратно на консоль
    cout << "\nВывод в файл закончен\n";
    cout << "Часть 4\n";
    cout << "Проверим ввод кирилицы с
консоли\nРезультат выведем на экран\n";
    cout << "Введите фамилию водителя, которого хотите
удалить\n";
    cin >> fio2;
    fio1=My_Translate(fio2);
    Erase_Vodit(fio1, L);
    fout.close();
    system("Pause");

return 0;
}
```

Файл DATA.TXT

```
12;МихНовичи;Орлов;Иванов;Петров;Давид;Жук;Орел
9;Каменка;Жуков;Давид;Сак;ЖУК;Васин
1;МИХНОВИЧИ;Харин;ПЕТРОВ;Жук;Семенов
222;МихновиЧИ;Жуков;Давид;Хорп;Харин;Жук;ПЕШКОВ
44;Роща;ПетроВ;Котов;Лука;Орлов;Жук
```

Файл ANSWER.TXT

```
Часть 1
Список до сортировки:
маршрут номер 12 название МихНовичи
```

```
водители: Орлов Иванов Петров Давид Жук Орел

маршрут номер 9 название Каменка
водители: Жуков Давид Сак ЖУК Васин

маршрут номер 1 название МИХНОВИЧИ
водители: Харин ПЕТРОВ Жук Семенов

маршрут номер 222 название МихновиЧИ
водители: Жуков Давид Хорп Харин Жук ПЕШКОВ

маршрут номер 44 название Роща
водители: ПетроВ Котов Лука Орлов Жук

Список после сортировки:
маршрут номер 9 название Каменка
водители: Жуков Давид Сак ЖУК Васин

маршрут номер 1 название МИХНОВИЧИ
водители: Харин ПЕТРОВ Жук Семенов

маршрут номер 12 название МихНовичи
водители: Орлов Иванов Петров Давид Жук Орел

маршрут номер 222 название МихновиЧИ
водители: Жуков Давид Хорп Харин Жук ПЕШКОВ

маршрут номер 44 название Роща
водители: ПетроВ Котов Лука Орлов Жук

Часть 2
Список всех водителей:
Васин Давид ЖУК Жуков Иванов Котов Лука Орел Орлов
ПЕТРОВ ПЕШКОВ Сак Семенов Харин Хорп

ПЕТРОВ не работает на следующих маршрутах:9 222

Зорин не работает на следующих маршрутах:9 1 12 222 44

Жук не работает на следующих маршрутах:нет таких
маршрутов

Список  водителей, которые работают на всех маршрутах:
ЖУК
```

```
Часть 3
Количество маршрутов в Михновичи = 3
В Михалово нет маршрутов

Удаление водителей
Список после удаления водителя Жук:
маршрут номер 9 название Каменка
водители: Жуков Давид Сак Васин

маршрут номер 1 название МИХНОВИЧИ
водители: Харин ПЕТРОВ Семенов

маршрут номер 12 название МихНовичи
водители: Орлов Иванов Петров Давид Орел

маршрут номер 222 название МихновиЧИ
водители: Жуков Давид Хорп Харин ПЕШКОВ

маршрут номер 44 название Роща
водители: ПетроВ Котов Лука Орлов

Нет водителя Клюев и список не изменился

Список после удаления водителя Давид:
маршрут номер 9 название Каменка
водители: Жуков Сак Васин

маршрут номер 1 название МИХНОВИЧИ
водители: Харин ПЕТРОВ Семенов

маршрут номер 12 название МихНовичи
водители: Орлов Иванов Петров Орел

маршрут номер 222 название МихновиЧИ
водители: Жуков Хорп Харин ПЕШКОВ

маршрут номер 44 название Роща
водители: ПетроВ Котов Лука Орлов

К части 2 дополнение после удаления
Список  водителей, которые работают на всех маршрутах:
нет таких водителей
```

Вид экрана после окончания работы