***Зеленковский 1 курс, 4 круппа***

***Main.cpp***

include <iostream>

#include "Country.h"

#include "WorldPopulation.h"

#include "CountryFeature.h"

#include "Program.h"

#include <string>

std::ifstream& operator >>(std::ifstream& in,CountryFeature &country\_feature);

std::ostream& operator <<(std::ostream & out,const CountryFeature &country);

int main() {

//Проверка работы метода ToString

std::map <int,long long int> country1;

country1.emplace(1960,9000000);

country1.emplace(1961,10000000);

Country country{"Belarus","Bel",country1};

std::string str\_out=country.ToString();

std::cout<<str\_out<<std::endl;

std::string new\_str=Country::ToString(country);

std::cout<<new\_str;

WorldPopulation countries\_data;

countries\_data.ReadCountryPopulationFromFile("C:\\Clion projects\\practice\\practice\_6\\world\_population\_1960-2018.csv");

countries\_data.CalcWorldPopulation();

countries\_data.WriteWorldPopulationToFile("C:\\Clion projects\\practice\\practice\_6\\output.csv");

//2. Создайте коллекцию стран, прочитав данные их исходного файла

std::vector<CountryFeature> CountriesFeatures;

CountryFeature country\_feature;

std::ifstream fin("C:\\Clion projects\\practice\\practice\_6\\countries.csv");

std::string header\_string;

getline(fin,header\_string);

if (!fin.is\_open()){

std::cout<<"Error";

return 0;

}

while(!fin.eof()){

fin >> country\_feature;

CountriesFeatures.push\_back(country\_feature);

}

return Program::Main();

}

std::ifstream& operator >>(std::ifstream& in,CountryFeature &country\_feature){

std::string str;

getline(in,str);

std::stringstream string\_stream(str);

std::string code,region,group;

getline(string\_stream,code,';');

country\_feature.set\_code(code);

getline(string\_stream,region,';');

country\_feature.set\_region(region);

getline(string\_stream,group,';');

country\_feature.set\_group(group);

return in;

}

//3.Реализуйте перегрузку operator<< для классов CountryFeature

std::ostream& operator <<(std::ostream& out,const CountryFeature& country){

out<<country.get\_code()<<";";

out<<country.get\_region()<<";";

out<<country.get\_group()<<";";

return out;

}

***Program.h***

#ifndef PRACTICE\_6\_PROGRAM\_H

#define PRACTICE\_6\_PROGRAM\_H

#include <string>

#include "WorldCountry.h"

#include "WorldPopulation.h"

#include "CountryFeature.h"

#include "WorldRegionPopulation!.h"

#include <vector>

class Program{

public:

static int Main();

std::vector<WorldCountry> countries\_data;

// 5,12 Создайте функцию (ReadCountryPopulationFromFile(string file) – чтение из файла в vector<WorldCountry>) –

// функцию-член класса Program, в которой на основе коллекций, содержащих объекты классов Country, CountryFeature

// создать коллекцию объектов класса WorldCountry.

void ReadCountryPopulationFromFile(std::string, std::string);

//6. Разработайте метод выбора стран из коллекции (аргументом будет уровень дохода страны),

//содержащей объекты класса WorldCountry, удовлетворяющих условиям:

//доход страны = High income (или = Low income, или = Lower middle income, или = Upper middle income).

std::vector<WorldCountry> ChooseByIncomeGroup (std::string income\_group);

//8. Разработайте метод формирование коллекции стран, принадлежащих какому-то региону (аргументом будет регион).

std::vector<WorldCountry> ChooseByRegion (std::string region);

//11,12 Для класса Program создайте метод void WriteWorldRegionPopulationToFile(string file) –

// запись в файл статистики мироворого населения по годам для каждого региона, в формате:

void WriteWorldRegionPopulationToFile(std::string file\_name);

private:

WorldPopulation world\_population;

};

#endif //PRACTICE\_6\_PROGRAM\_H

***Program.cpp***

#include "Program.h"

#include "WorldRegionPopulation!.h"

void Program::ReadCountryPopulationFromFile(std::string file\_name1="C:\\Clion projects\\practice\\practice\_6\\countries.csv",

std::string file\_name2="C:\\Clion projects\\practice\\practice\_6\\world\_population\_1960-2018.csv";){

//Чтение из первого файла вектора <CountryFeature>

CountryFeature country\_feature;

country\_feature.ReadCountryFeatureFromFile(file\_name1);

std::vector<CountryFeature> CountriesFeatures=country\_feature.get\_counties\_feature();

//Чтение из второго файла данных вектора <Country> (для получение country\_name, year\_population)

WorldPopulation world\_population;

world\_population.ReadCountryPopulationFromFile(file\_name2);

//Создание нового вектора на основе двух предыдущих

for(int i=0;i<world\_population.countries.size();i++){

WorldCountry worldCountry(world\_population.countries[i].country\_name,CountriesFeatures[i],world\_population.countries[i].year\_population);

countries\_data.push\_back(worldCountry);

}

}

std::vector<WorldCountry> Program::ChooseByIncomeGroup (std::string income\_group){

std::vector<WorldCountry> countriesByIncomeGroup;

for(const WorldCountry& worldCountry:countries\_data){

if(worldCountry.get\_feature().get\_group()==income\_group){

countriesByIncomeGroup.push\_back(worldCountry);

}

}

return countriesByIncomeGroup;

}

std::vector<WorldCountry> Program::ChooseByRegion (std::string region){

std::vector<WorldCountry> countriesByRegion;

for(const WorldCountry& worldCountry:countries\_data){

if(worldCountry.get\_feature().get\_region()==region){

countriesByRegion.push\_back(worldCountry);

}

}

return countriesByRegion;

}

void Program::WriteWorldRegionPopulationToFile(std::string file\_name="C:\\Clion projects\\practice\\practice\_6\\output\_7.csv"){

std::ofstream fout(file\_name);

fout<<"Region;";

for(int i=1960;i<=2018;i++){

fout<<i<<";";

}

fout<<"\n";

CountryFeature country\_feature;

Program program;

WorldRegionPopulation region\_population;

std::set<std::string> regions= country\_feature.GetUniqueRegions();

for(std::string region\_name:regions){

fout<<region\_name<<";";

region\_population.set\_region(region\_name);

for(int i=1960;i<=2018;i++){

region\_population.set\_year\_population(std::pair<int,long long>(i,0));

}

region\_population.CalcWorldRegionPopulation();

for(int i=1960;i<=2018;i++) {

fout << region\_population.get\_year\_population()[i]<<";";

}

fout<<std::endl;

}

}

***WORLDREGIONPOPULATION.H***

#ifndef PRACTICE\_6\_WORLDREGIONPOPULATION\_H

#define PRACTICE\_6\_WORLDREGIONPOPULATION\_H

#include <string>

#include <map>

#include <vector>

#include "Program.h"

//9. Создайте класс WorldRegionPopulation с двумя полями:

//region, map<int,long long int> year\_population.

class WorldRegionPopulation {

private:

std::string region;

std::map<int, long long int> year\_population;

public:

void set\_region(std::string \_region){region=\_region;};

void set\_year\_population(std::pair<int,long long> pair){year\_population.insert(pair);};

//10. Для класса WorldRegionPopulation создайте метод (функцию-член) void CalcWorldRegionPopulation() –

// вычисление населения всех регионах по годам, используйте коллекцию из п.8.

void CalcWorldRegionPopulation();

void WorldRegionPopulation(Program& \_program, const std::string& \_region):program(\_program),region(\_region){

for(int i=1960;i<=2018;i++){

year\_population.insert(std::pair<int,long long>(i,0));

}

CalcWorldRegionPopulation();

}

std::map<int, long long> get\_year\_population() { return this->year\_population;};

private:

Program program;

};

#endif //PRACTICE\_6\_WORLDREGIONPOPULATION\_H

***WORLDREGIONPOPULATION.CPP***

#include "WorldRegionPopulation.h"

void WorldRegionPopulation::CalcWorldRegionPopulation() {

Program program;

std::vector<WorldCountry> countries\_by\_region = program.ChooseByRegion(region);

for (const WorldCountry &country:countries\_by\_region) {

for (auto year\_population\_pair:country.get\_year\_population()) {

year\_population[year\_population\_pair.first] += year\_population\_pair.second;

}

}

}

***WORLDPOPULATION.H***

#ifndef PRACTICE\_6\_WORLDPOPULATION\_H

#define PRACTICE\_6\_WORLDPOPULATION\_H

#include <iostream>

#include <vector>

#include <map>

#include "Country.h"

#include <fstream>

#include <sstream>

struct WorldPopulation{

std::vector<Country> countries;

std::map<int,long long int> year\_population;

void ReadCountryPopulationFromFile(std::string file\_name);

void CalcWorldPopulation();

void WriteWorldPopulationToFile(std::string file\_name);

};

#endif //PRACTICE\_6\_WORLDPOPULATION\_H

***WORLDPOPULATION.CPP***

#include "WorldPopulation.h"

void WorldPopulation::ReadCountryPopulationFromFile(std::string file\_name) {

std::ifstream in\_file(file\_name);

if (!in\_file.is\_open()) {

std::cout << "Error file open\n";

return;

};

std::string header\_string;

std::getline(in\_file, header\_string);

while (!in\_file.eof()) {

std::string county\_str;

std::getline(in\_file, county\_str);

std::stringstream in(county\_str);

Country country\_data;

std::getline(in, country\_data.country\_name, ';');

std::getline(in, country\_data.country\_code, ';');

for (int i = 1960; i <= 2018; i++) {

long long int population = 0;

std::string population\_str;

std::getline(in, population\_str, ';');

if (population\_str.length() == 0) {

country\_data.year\_population[i] = population;

} else {

population = std::stoll(population\_str);

country\_data.year\_population.insert(std::make\_pair(i,population));

}

}

this->countries.push\_back(country\_data);

}

}

void WorldPopulation::CalcWorldPopulation(){

for(Country country:countries){

for(int i=1960;i<=2018;i++){

this->year\_population[i]+=country.year\_population[i];

}

}

}

void WorldPopulation::WriteWorldPopulationToFile(std::string file\_name){

std::ofstream out(file\_name);

for(int i=1960;i<=2018;i++){

out<<i<<";";

}

out<<"\n";

for(int i=1960;i<=2018;i++){

out<<this->year\_population[i]<<";";

}

***WORLDCOUNTRY.H***

#ifndef PRACTICE\_6\_WORLDCOUNTRY\_H

#define PRACTICE\_6\_WORLDCOUNTRY\_H

#include "CountryFeature.h"

#include <iostream>

#include <map>

//4. По аналогии со структурой Country создайте класс WorldCountry c полями:

// string сountry\_name, CountryFeature country\_feature, map<int, long long int> year\_population.

class WorldCountry{

private:

std::string country\_name;

CountryFeature country\_feature;

std::map<int, long long int> year\_population;

public:

WorldCountry(const std::string& \_country\_name, const CountryFeature& \_country\_feature,const std::map<int, long long int>& \_year\_population):

country\_name(\_country\_name),country\_feature(\_country\_feature),year\_population(\_year\_population){}

std::string get\_name() const {return this->country\_name;};

CountryFeature get\_feature() const {return this->country\_feature;};

std::map<int, long long int> get\_year\_population() const {return this->year\_population;};

};

#endif //PRACTICE\_6\_WORLDCOUNTRY\_H

**COUNTRYFEATURE.H**

#ifndef PRACTICE\_6\_COUNTRYFEATURE\_H

#define PRACTICE\_6\_COUNTRYFEATURE\_H

#include <iostream>

#include <vector>

#include <fstream>

#include <sstream>

#include <set>

//1.Создайте класс CountryFeature с тремя полями:

//string country\_code, string region, string income\_group

class CountryFeature {

private:

std::string country\_code;

std::string region;

std::string income\_group;

std::vector<CountryFeature> coutries\_feature;

public:

std::string get\_code() const {return this->country\_code;};

std::string get\_region() const { return this->region;};

std::string get\_group() const {return this->income\_group;};

std::vector<CountryFeature> get\_counties\_feature() const{return this->coutries\_feature;};

void set\_code (std::string code){country\_code=code;}

void set\_region (std::string str\_region){region=str\_region;}

void set\_group (std::string group){income\_group=group;}

void ReadCountryFeatureFromFile(std::string);

//7. Разработайте метод формирования коллекции уникальных регионов из коллекции объектов CountryFeature.

std::set<std::string> GetUniqueRegions();

};

std::ifstream& operator >>(std::ifstream& in,CountryFeature &country\_feature);

#endif //PRACTICE\_6\_COUNTRYFEATURE\_H

**COUNTRYFEATURE.CPP**

#include "CountryFeature.h"

std::ifstream& operator >>(std::ifstream& in,CountryFeature &country\_feature){

std::string str;

getline(in,str);

std::stringstream string\_stream(str);

std::string code,region,group;

getline(string\_stream,code,';');

country\_feature.set\_code(code);

getline(string\_stream,region,';');

country\_feature.set\_region(region);

getline(string\_stream,group,';');

country\_feature.set\_group(group);

return in;

}

void CountryFeature::ReadCountryFeatureFromFile(std::string file\_name){

CountryFeature country\_feature;

std::ifstream fin(file\_name);

std::string header\_string;

getline(fin,header\_string);

if (!fin.is\_open()){

std::cout<<"Error";

}

while(!fin.eof()){

fin >> country\_feature;

coutries\_feature.push\_back(country\_feature);

}

}

std::set<std::string> CountryFeature::GetUniqueRegions(){

std::set<std::string> unique\_regions;

this->ReadCountryFeatureFromFile("C:\\Clion projects\\practice\\practice\_6\\countries.csv");

std::vector<CountryFeature> courtries\_feature=this->get\_counties\_feature();

for(const CountryFeature& countryFeature:courtries\_feature){

unique\_regions.insert(countryFeature.get\_region());

}

return unique\_regions;

}

***COUNTRY.H***

#ifndef PRACTICE\_6\_COUNTRY\_H

#define PRACTICE\_6\_COUNTRY\_H

#include <map>

#include <iostream>

#include <iterator>

struct Country{

std::string country\_name;

std::string country\_code;

std::map<int,long long int> year\_population;

std::string ToString();

static std::string ToString(Country country);

};

std::ostream & operator <<(std::ostream & out,const Country &country){

std::string str=country.ToString();

out<<str;

return out;

}

#endif //PRACTICE\_6\_COUNTRY\_H

***COUNTRY.CPP***

#include "Country.h"

std::string Country::ToString(){

std::string str=this->country\_name+"("+this->country\_code+"): ";

for(int i= 1960;i<=1960+this->year\_population.size()-1;i++) {

std::map<int,long long>::iterator it=this->year\_population.find(i);

str+=std::to\_string(it->second) + "[" + std::to\_string(it->first)+"], ";

}

if(\*(str.end()-2)==','){

str.erase(str.length()-2,2);

}

return str;

}

std::string Country::ToString(Country country) {

std::string str=country.country\_name+"("+country.country\_code+"): ";

for(int i= 1960;i<=1960+country.year\_population.size()-1;i++) {

std::map<int,long long>::iterator it=country.year\_population.find(i);

str+=std::to\_string(it->second) + "[" + std::to\_string(it->first)+"], ";

}

if(\*(str.end()-2)==','){

str.erase(str.length()-2,2);

}

return str;

}