Задание по теме «НАСЛЕДОВАНИЕ»

За отчетный период проводится **три**  чемпионата по спорту. Информация о выступлении спортсменов накапливается во входном файле. Строки файла имеют следующую структуру

Файл ПОЯСНЕНИЯ

|  |
| --- |
| AAA 6 1  BBB 10 2  CCC 5 3  AAA 7 3  BBB 8 1  CCC 9 1  AAA 9 2  BBB 9 3 |

Фамилия результат №чемпионата(протокола)

К сожалению, при передаче досье на спортсменов в аналитический центр протоколы были перемешаны и размещены **в   
 multimap<string,** **имя\_базового\_класса\*>m\_s.**  Номер протокола использовался для выбора типа класса для создании соответствующего объекта. Например, по второму протоколу создан объект класса «Спортсмен\_2», хранящий фамилию BBB и результат 10 баллов и в контейнере. Для ссылки на объект используем указатель базового типа.

Разработать следующую иерархию классов:

- базовый класс «Спортсмен» с полем «Фамилия». У всех ( от 4 до 5 ) спортсменов различные фамилии.

- производный класс «Спортсмен\_1», в котором фиксируется «Результат» **после первого** чемпионата. Результат сохраняется как целое число в диапазоне [0..10].

- производный класс «Спортсмен\_2», в котором фиксируется «Результат**» после второго** чемпионата;

- производный класс «Спортсмен\_3», в котором фиксируется «Результат» **после третьего** чемпионата.

Необходимо, выбирая информацию из построенного контейнера, распознать (динамически) тип объекта и распределить ее в трех **map** отдельно для каждого чемпионата.

Для спортсменов, участвовавших **во всех 3-х чемпионатах**, найти:  
[1] тех, чьи результаты растут, [2] тех, кто стабильно показывает высокие результаты (9 или 10).

Вычислить рейтинг (среднее значение) для каждого чемпионата (можно не создавая отдельные map для 1 и 2 чемпионатов).

**Пример работы по созданию и обработке похожих классов**

#include <iostream>

#include <string>

#include <map>

using namespace std;

class Man {

public:

string name;

Man(string s="") { name = s; }

virtual void f() {} // виртуальная фукция добавлена для полиморфизма иерархии классов

};

class Man1 :public Man {

public:

int res1;

Man1(string s, int r1) :Man(s) { res1 = r1; }

virtual void f() {}

};

class Man2 :public Man {

public:

int res2;

Man2(string s, int r2) :Man(s) { res2 = r2; }

virtual void f() {}

};

int main() {

multimap<string, Man\*> sport;

Man1 c("CCC", 7);

Man1 a("AAA", 77);

Man2 b("AAA", 777);

sport.insert(make\_pair("AAA", &b));

sport.insert(make\_pair("CCC", &c));

sport.insert(make\_pair("AAA", &a));

//Проверка контейнера для объекта класса Man2

multimap<string, Man\*>::iterator it = sport.begin();

cout << (\*it).first << ' ' << dynamic\_cast<Man2\*>((\*it).second)->res2 << endl;

}