שם: אבי מישייב

תז: 308254549

תשובה לשאלה:

הבאג הלוגי הוא שלא ניתן לשנות את שדה הid חוץ מביצירית האוביקט דרך הבנאי. לכן כש yaniv katan נוסף למערך לא הייתה אפשרות להוסיף את הid שלו. כאשר התוכנית פנתה לפונקיה שמחפשת לפי תז היא לא מצאה את yaniv katan למרות שהנתונים בmain זהים.

כדי לתקן זאת הייתי מוריד את הקבוע מהid והופך את התכונה לprotected ומוסיף הרשאת גישה לliga.

Main.cpp:

#include "Player.h"

#include "Liga.h"

void main()

{

Player::printBestPlayer();

Player p1("Tal", "Ben-Haim", 22514753);

cout << "Player p1 " << p1;

Player p2("Yossi", "Ben-Ayun", 33158420);

Player p3(p2);

Player \*p4 = new Player(p3);

p3.set\_name("Eran", "Zehavi");

p3.set\_points(20);

p2.printBestPlayer();

p1 += 20;

p2 += 40;

if (p1 == p3)

cout << "p1 and p3 have the same number of points:" << p1.getCareerPts() << endl;

if (p1 != p2)

cout << "p1 and p2 doesn't have the same number of points\n";

p3.set\_last\_name("very very long last name");

p4->set\_name("Miki", "Berkovich");

++(\*p4);

p4->printBestPlayer();

Player p5;

Player arr[5] = { p1,p2,p3,\*p4,p5 };

Liga li1(arr, 5, 11);

Liga li2(10);

li2.print();

li2.addPlayer("Yaniv", "katan", 35822669);

int index = li2.getPerson(35822669);

li2.print\_player(index);

cout << "\nNumbers of players in all liga in Israel is " << li1.numPlayers() << endl;

li2 += (li1);

li2.print();

delete p4;

system("pause");

}

Player.h:

#ifndef \_\_Player\_H

#define \_\_Player\_H

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <cstring>

using namespace std;

class Player {

char \*name;

char family[30];

int numOfVest;

const int id;

int careerPts;

static int counter;

static Player \*BestPlayer;

friend class Liga;

friend ostream& operator<<(ostream& out, const Player& p);

public:

//set's

void setName(const char \*name);

void setFamily(const char \*family);

void setNumOfVest(int numOfVest);

void setCareerPts(int careerPts);

//get's

const char\* getName() const { return name; };

const char\* getFamily() const { return &family[0]; };

int getNumOfVest() const { return numOfVest; };

int getCareerPts() const { return careerPts; };

//constructor's

Player(char \*name, char family[30], int id);

Player();

Player(const Player &o);

//destructor's

~Player();

//opertor's

const Player& operator=(const Player& o);

const Player& operator+=(int num);

bool operator==(const Player& o) const;

bool operator!=(const Player& o) const;

void operator++();

//Best Player - static

void setBestPlayer();

static void printBestPlayer();

//methods

void print() const;

void set\_name(char \*name, char\* family);

void set\_points(int pts);

void set\_last\_name(char \*family) { setFamily(family); };

};

#endif

Player.cpp:

#include "Player.h"

int Player::counter = 10;

Player\* Player::BestPlayer = NULL;

void Player::setName(const char \*name) {

delete[] this->name;

this->name = new char[strlen(name) + 1];

strcpy(this->name, name);

}

void Player::setFamily(const char \*family) {

strcpy(this->family, family);

}

void Player::setNumOfVest(int numOfVest) {

this->numOfVest = numOfVest;

}

void Player::setCareerPts(int careerPts) {

this->careerPts = careerPts;

}

Player::Player(char \*name, char family[30], int id) : id(id) {

this->name = NULL;

setName(name);

setFamily(family);

setCareerPts(0);

setNumOfVest(counter++);

setBestPlayer();

}

Player::Player() : id(0) {

this->name = NULL;

setName("Default");

setFamily("Name");

setNumOfVest(counter++);

setCareerPts(0);

setBestPlayer();

}

Player::~Player(){

delete[] name;

}

Player::Player(const Player &o):id(o.id) {

name = NULL;

\*this = o;

setBestPlayer();

}

const Player& Player::operator=(const Player& o) {

if (this != &o) {

name = NULL;

setName(o.getName());

setFamily(o.getFamily());

this->numOfVest = o.numOfVest;

this->careerPts = o.careerPts;

}

return \*this;

}

void Player::setBestPlayer() {

if (!BestPlayer)

BestPlayer = this;

else if (BestPlayer->careerPts < this->careerPts ) {

BestPlayer = this;

}

}

void Player::printBestPlayer() {

if (BestPlayer == NULL)

cout << "there are no players\n";

else {

cout << "the best player is: ";

BestPlayer->print();

}

}

void Player::print() const {

cout << "Player name: " << name << " " << family << " shirt number: " << numOfVest << " number of points: " << careerPts << endl;

}

void Player::set\_name(char \*name, char\* family) {

setName(name);

setFamily(family);

}

void Player::set\_points(int pts) {

setCareerPts(pts);

setBestPlayer();

}

ostream& operator<<(ostream& out, const Player& p)

{

out << "Player name: " << p.getName() << " " << p.getFamily() << " shirt number: " << p.getNumOfVest() << " number of points: " << p.getCareerPts() << endl;

return out;

}

const Player& Player::operator+=(int num) {

careerPts += num;

setBestPlayer();

return \*this;

}

bool Player::operator==(const Player& o) const {

if (careerPts == o.getCareerPts())

return true;

}

bool Player::operator!=(const Player& o) const {

if (careerPts != o.getCareerPts())

return true;

}

void Player::operator++() {

careerPts++;

setBestPlayer();

}

Liga.h:

#ifndef \_\_Liga\_H

#define \_\_Liga\_H

#include "Player.h"

class Liga {

Player \*ligaPlayers;

int maxLigaPlayers;

static int numOfLeaguesPlayers;

int numExistPlayers;

public:

//constructor's

Liga(int maxLigaPlayers);

Liga(const Player \*other, int size, int maxLigaPlayers);

Liga(const Liga& o);

//destructor

~Liga();

//operator's

const Liga& operator=(const Liga& o);

const Liga& operator+=(const Liga& o);

//methods

void setPoints(char family[30], int pts);

void addPlayer(char \*name, char \*family, int id);

int getPerson(int id) const;

void print() const;

void print\_player(int i) const;

int numPlayers() const { return numOfLeaguesPlayers; };

};

#endif

Liga.cpp:

#include "Liga.h"

int Liga::numOfLeaguesPlayers = 0;

Liga::Liga(int maxLigaPlayers) {

ligaPlayers = new Player[maxLigaPlayers];

this->maxLigaPlayers = maxLigaPlayers;

}

Liga::Liga(const Player \*other, int size, int maxLigaPlayers) {

size <= maxLigaPlayers ? numExistPlayers = size : numExistPlayers = maxLigaPlayers;

this->ligaPlayers = new Player[numExistPlayers];

this->maxLigaPlayers = maxLigaPlayers;

for (int i = 0; i < size; i++) {

if (i == maxLigaPlayers)

break;

ligaPlayers[i] = other[i];

numOfLeaguesPlayers++;

}

}

Liga::~Liga() {

delete[] ligaPlayers;

}

Liga::Liga(const Liga& o) {

ligaPlayers = NULL;

\*this = o;

}

const Liga& Liga::operator=(const Liga& o) {

if (this != &o) {

maxLigaPlayers = o.maxLigaPlayers;

ligaPlayers = new Player[maxLigaPlayers];

for (int i = 0; i < maxLigaPlayers; i++) {

ligaPlayers[i] = o.ligaPlayers[i];

numOfLeaguesPlayers++;

}

}

return \*this;

}

void Liga::setPoints(char family[30], int pts) {

int i = 0;

while (strcmp(ligaPlayers[i].getFamily(), family)&&i<maxLigaPlayers)

i++;

if (i == maxLigaPlayers)

cout << "there is no player with this family name"<<endl;

ligaPlayers[i].setCareerPts(pts);

}

void Liga::addPlayer(char \*name, char \*family, int id) {

Player \*tmp = ligaPlayers;

if (numExistPlayers == maxLigaPlayers)

return;

ligaPlayers = new Player[numExistPlayers + 1];

for(int i=0;i<numExistPlayers-1;i++)

ligaPlayers[i] = tmp[i];

delete[] tmp;

ligaPlayers[numExistPlayers].setName(name);

ligaPlayers[numExistPlayers].setFamily(family);

numOfLeaguesPlayers++;

numExistPlayers++;

}

int Liga::getPerson(int id) const {

int i = 0;

while ((ligaPlayers[i].id != id)) {

i++;

if (i == maxLigaPlayers) {

return -1;

}

}

return i;

}

void Liga::print() const {

//if (!strcmp(ligaPlayers[0].getName(), "Default"))

if(ligaPlayers==NULL)

cout << "There are nnot players in this Liga";

else {

cout << "Liga's Players:\n";

for (int i = 0; i < numExistPlayers; i++)

cout << ligaPlayers[i];

}

}

const Liga& Liga::operator+=(const Liga& o) {

for (int i = 0; (i < o.numExistPlayers) && (i < maxLigaPlayers); i++) {

ligaPlayers[numExistPlayers++] = o.ligaPlayers[i];

}

return \*this;

}

void Liga::print\_player(int i) const{

if (i = -1)

cout<<"no such player";

else

cout << ligaPlayers[i];

}

