ДЗ к 11.04.2023

0) придумать примеры, демонстрирующие разницу между префиксным и постфиксным инкрементом на примере класса

int main()

{

BowlingBall b(5, "white");

b++.print();

b.print();

BowlingBall b2(3, "yellow");

(++b2).print();

return 0;

}

1) в классе BowlingBall переопределить сравнение "<" по весу и сравнение "==" и по весу, и по цвету

bool operator<(BowlingBall other1) {

return weight < other1.weight;

}

bool operator==(BowlingBall other1) {

return (weight == other1.weight) && (color == other1.color);

}

int BowlingBall::count = 0;

int main()

{

BowlingBall b(5, "white");

b++.print();

b.print();

BowlingBall b2(3, "yellow");

(++b2).print();

cout << (b2 < b) << endl;

cout << (b2 == b) << endl;

BowlingBall b3(3, "yellow");

cout << (b2 == b3) << endl;

return 0;

}

2) использовать перегруженное в задании 1 сравнение для сортировки шаров по весу

int main()

{

int n;

cin >> n;

BowlingBall\* balls = new BowlingBall[n];

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

int wes;

string cvet;

cin >> wes >> cvet;

balls[i].setParams(wes, cvet);

}

for (int i = 1; i < n; i++) {

for (int j = i; j > 0 && (balls[j] < balls[j - 1]); j--) {

swap(balls[j - 1], balls[j]);

}

}

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

balls[i].print();

cout << endl;

}

}

3)создать статический метод (либо внешнюю функцию) для сравнения двух шаров, такую, чтобы с её помощью сортировать шары по убыванию веса, а при равенстве по цвету лексикографически. Примерная сигнатура функции:

static bool isGreater(const BowlingBall &x, const BowlingBall &y)

static bool isGreater(const BowlingBall& x, const BowlingBall& y) {

if (x.weight > y.weight) {

return true;

}

else if (x.weight < y.weight) {

return false;

}

else {

return x.color > y.color;

}

}

// из int main():

for (int i = 1; i < n; i++) {

for (int j = i; j > 0 && BowlingBall :: isGreater(balls[j], balls[j - 1]); j--) {

swap(balls[j - 1], balls[j]);

}

}

4) Подумать, что должны "знать" и "уметь" прямоугольники, чтобы с помощью класса можно было: создать прямоугольник по длине и ширине, создать прямоугольник по левой верхней и правой нижней точке, создать прямоугольник по левой верхней точке, длине и ширине (тут есть некоторый подвох), распечатать сведения о прямоугольнике, распечатать все 4 точки прямоугольника, вернуть длину, вернуть ширину, установить новую длину, установить новую ширину, "переместить" прямоугольник: размеры остаются прежними, точки сдвигаются (задаётся новый левый верхний угол). Реализовать такой класс. Подумать, какие ещё возможности можно добавить.

class Rectangle {

int x1, y1;

int x2, y2;

double a, b;

public:

Rectangle(int a, int b) {

this->a = a;

this->b = b;

y1 = b;

x2 = a;

x1 = 0;

y2 = 0;

}

Rectangle(int x1, int y1, int x2, int y2) {

a = x2 - x1;

b = y1 - y2;

this->x1 = x1;

this->x2 = x2;

this->y1 = y1;

this->y2 = y2;

}

Rectangle(int x1, int y1, double a, int b) {

this->x1 = x1;

this->y1 = y1;

this->a = a;

this->b = b;

x2 = a + x1;

y2 = y1 - b;

}

void print() {

cout << "(" << x1 << ", " << y1 << "), ";

cout << "(" << x2 << ", " << y2 << "), ";

cout << a << ", " << b;

}

void print\_points() {

cout << "(" << x1 << ", " << y2 << "), "; //левая нижняя

cout << "(" << x1 << ", " << y1 << "), "; //левая верхняя

cout << "(" << x2 << ", " << y1 << "), "; //правая верхняя

cout << "(" << x2 << ", " << y2 << ") "; //правая нижняя

}

int getWidth() {

return b;

}

int getLength() {

return a;

}

void newLength(int c) {

a = c;

if (c < a) {

x2 = a - x1;

}

else {

x2 = a + x1;

}

}

void newWidth(int c) {

if (c < b) {

b = b - c;

}

else {

b = c;

}

y1 = b + y2;

}

void Moving(int x, int y) {

y2 = y2 + (y - y1);

x2 = x2 + (x - x1);

x1 = x;

y1 = y;

}

int Square() {

return a \* b;

}

int Perimeter() {

return 2 \* a + 2 \* b;

}

double Diagonal() {

return sqrt((x2 - x1) \* (x2 - x1) + (y2 - y1) \* (y2 - y1));

}

};