#include <iostream>

#include <string>

using namespace std;

//1-1

class Book {

string title;

int price, pages;

public:

Book(string title = "", int price = 0, int pages = 0) {

this->title = title;

this->price = price;

this->pages = pages;

}

Book& operator+=(int n) {

price += n;

return \*this;

}

bool operator-=(int price) {

return this->price -= price;

}

void show() {

cout << title << " " << price << "원 " << pages << " 페이지" << endl;

}

};

int main() {

Book a("청춘", 20000, 300), b("미래", 30000, 500);

a += 500;

b -= 500;

a.show();

b.show();

return 0;

}



#include <iostream>

#include <string>

using namespace std;

//1-2

class Book;

Book& operator+=(Book& b, int price);

Book& operator-=(Book& b, int price);

class Book {

string title;

int price;

int pages;

public:

Book(string t = "", int p = 0, int pg = 0) : title(t), price(p), pages(pg) {}

void show() const {

cout << title <<" " << price << "원 " << pages << "페이지" << endl;

}

string getTitle() const { return title; }

friend Book& operator+=(Book& b, int price);

friend Book& operator-=(Book& b, int price);

};

Book& operator+=(Book& b, int price) {

b.price += price;

return b;

}

Book& operator-=(Book& b, int price) {

b.price -= price;

return b;

}

int main() {

Book a("청춘", 20000, 300);

Book b("미래", 30000, 500);

a += 500;

b -= 500;

a.show();

b.show();

return 0;

}



#include <iostream>

#include <string>

using namespace std;

//3

class Book {

string title;

int price;

int pages;

public:

Book(const string& t = "", int p = 0, int pg = 0)

: title(t), price(p), pages(pg) {

}

void show() const {

cout << title << " " << price << "원 " << pages << " 페이지" << endl;

}

string getTitle() const { return title; }

bool operator!() const {

return price == 0;

}

};

int main() {

Book book("벼룩시장", 0, 50);

if (!book) {

cout << "공짜다" << endl;

}

return 0;

}



#include <iostream>

#include <string>

using namespace std;

//5-1

class Color {

int red;

int green;

int blue;

public:

Color() : Color(0, 0, 0) {}

Color(int r, int g, int b) : red(r), green(g), blue(b) {}

void show() const {

cout << red << ' ' << green << ' ' << blue << endl;

}

Color operator+(const Color& c) const {

return Color(red + c.red, green + c.green, blue + c.blue);

}

bool operator==(const Color& c) const {

return red == c.red && green == c.green && blue == c.blue;

}

};

int main() {

Color red(255, 0, 0), blue(0, 0, 255), c;

c = red + blue;

c.show();

Color fuchsia(255, 0, 255);

if (c == fuchsia)

cout << "보라색 맞음" << endl;

else

cout << "보라색 아님" << endl;

return 0;

}



using namespace std;

//5-2

class Color;

Color operator+(const Color& c1, const Color& c2);

bool operator==(const Color& c1, const Color& c2);

class Color {

int red;

int green;

int blue;

public:

Color() : Color(0, 0, 0) {}

Color(int r, int g, int b) : red(r), green(g), blue(b) {}

void show() const {

cout << red << ' ' << green << ' ' << blue << endl;

}

friend Color operator+(const Color& c1, const Color& c2);

friend bool operator==(const Color& c1, const Color& c2);

};

Color operator+(const Color& c1, const Color& c2) {

return Color(c1.red + c2.red, c1.green + c2.green, c1.blue + c2.blue);

}

bool operator==(const Color& c1, const Color& c2) {

return (c1.red == c2.red && c1.green == c2.green && c1.blue == c2.blue);

}

int main() {

Color red(255, 0, 0), blue(0, 0, 255), c;

c = red + blue;

c.show();

Color fuchsia(255, 0, 255);

if (c == fuchsia)

cout << "보라색 맞음" << endl;

else

cout << "보라색 아님" << endl;

return 0;

}



#include <iostream>

using namespace std;

//7-1

class Matrix {

int m[4];

public:

Matrix() : Matrix(0, 0, 0, 0) {}

Matrix(int m1, int m2, int m3, int m4) {

m[0] = m1; m[1] = m2; m[2] = m3; m[3] = m4;

}

void show() const {

cout << "Matrix = { ";

for (int i = 0; i < 4; i++)

cout << m[i] << ' ';

cout << "}" << endl;

}

void operator>>(int\* mat) const {

for (int i = 0; i < 4; i++)

mat[i] = m[i];

}

Matrix& operator<<(const int\* mat) {

for (int i = 0; i < 4; i++)

m[i] = mat[i];

return \*this;

}

};

int main() {

Matrix a(4, 3, 2, 1), b;

int x[4], y[4] = { 1, 2, 3, 4 };

a >> x;

b << y;

for (int i = 0; i < 4; i++) cout << x[i] << ' ';

cout << endl;

b.show();

return 0;

}



#include <iostream>

using namespace std;

//7-2

class Matrix;

void operator>>(const Matrix& mat1, int mat2[4]);

Matrix& operator<<(Matrix& mat1, const int mat2[4]);

class Matrix {

int m[4];

public:

Matrix() : Matrix(0, 0, 0, 0) {}

Matrix(int m1, int m2, int m3, int m4) {

m[0] = m1; m[1] = m2; m[2] = m3; m[3] = m4;

}

void show() const {

cout << "Matrix = { ";

for (int i = 0; i < 4; i++)

cout << m[i] << ' ';

cout << "}" << endl;

}

friend void operator>>(const Matrix& mat1, int mat2[4]);

friend Matrix& operator<<(Matrix& mat1, const int mat2[4]);

};

void operator>>(const Matrix& mat1, int mat2[4]) {

for (int i = 0; i < 4; i++)

mat2[i] = mat1.m[i];

}

Matrix& operator<<(Matrix& mat1, const int mat2[4]) {

for (int i = 0; i < 4; i++)

mat1.m[i] = mat2[i];

return mat1;

}

int main() {

Matrix a(4, 3, 2, 1), b;

int x[4], y[4] = { 1, 2, 3, 4 };

a >> x;

b << y;

for (int i = 0; i < 4; i++)

cout << x[i] << ' ';

cout << endl;

b.show();

return 0;

}

s

#include <iostream>

using namespace std;

//9

class Circle;

Circle operator+(int r, const Circle& c);

class Circle {

int radius;

public:

Circle(int r = 0) : radius(r) {}

void show() const {

cout << "radius = " << radius << " 인 원" << endl;

}

friend Circle operator+(int r, const Circle& c);

};

Circle operator+(int r, const Circle& c) {

return Circle(r + c.radius);

}

int main() {

Circle a(5), b(4);

b = 1 + a;

a.show();

b.show();

return 0;

}



#include <iostream>

using namespace std;

//11

class Stack {

int s[100];

int top;

public:

Stack() : top(-1) {}

bool operator!() const {

return top == -1;

}

Stack& operator<<(int x) {

if (top < 99) s[++top] = x;

return \*this;

}

Stack& operator>>(int& x) {

if (top >= 0) x = s[top--];

return \*this;

}

};

int main() {

Stack stack;

stack << 3 << 5 << 10;

while (true) {

if (!stack) break;

int x;

stack >> x;

cout << x << ' ';

}

cout << endl;

return 0;

}

