#include <iostream>

using namespace std;

int counter = 1;

class Rational {

private:

int n, d;

public:

Rational(int = 0, int = 1);

void printFraction();

double retFloating();

};

Rational::Rational(int n, int d) {

this -> n = n;

this -> d = d;

}

void Rational::printFraction() {

counter++;

if (counter > 0 && counter < 100) {

cout << n << "/" << d << " + ";

} else {

cout << n << "/" << d << " ";

}

}

double Rational::retFloating() {

return(float)n/d;

}

int main() {

double sum = 0;

for (int i = 1; i < 100; i++) {

Rational x(i, i+1);

x.printFraction();

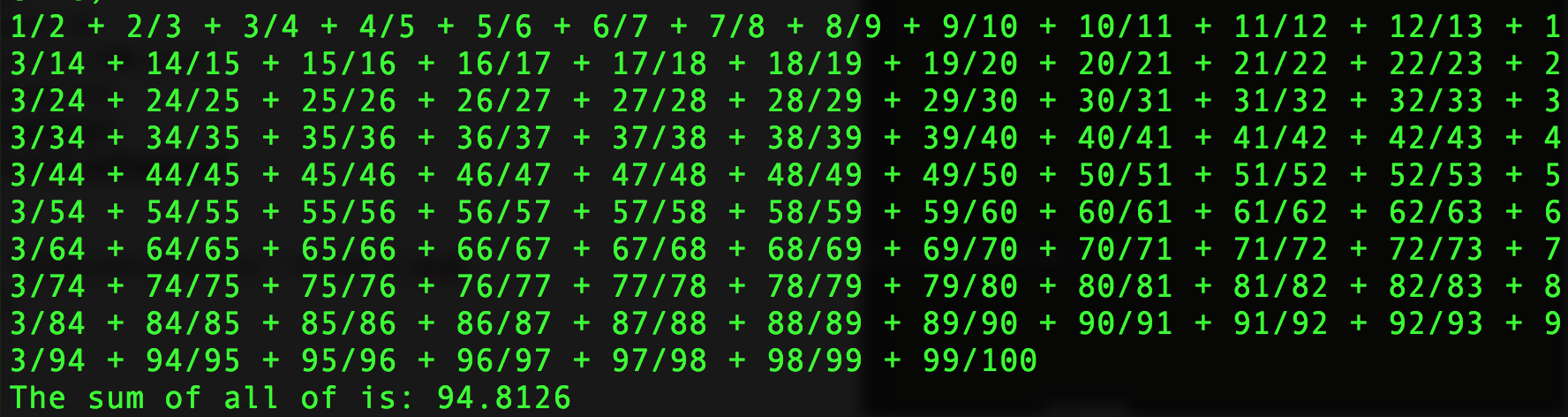
sum = sum+x.retFloating();

}

cout << "\nThe sum of all of is: " << sum << endl;

return 0;

}



#include <iostream>

using namespace std;

class Circle {

private:

double r;

public:

Circle() {

r = 1;

}

Circle(double newR) {

r = newR;

}

double getArea() {

return r \* r \* 3.14159;

}

double getR() {

return r;

}

Circle subtract(Circle &secondC) {

double r = r - secondC.getR();

return Circle(r);

}

int compareTo(Circle &secondC) {

Circle temp = subtract(secondC);

if (temp.getR() < 0) {

return -1;

} else if (temp.getR() == 0) {

return 0;

} else {

return 1;

}

}

friend bool operator < (Circle circle1, Circle circle2) {

return (circle1.compareTo(circle2) < 0);

}

friend bool operator <= (Circle circle1, Circle circle2) {

return (circle1.compareTo(circle2) <= 0);

}

friend bool operator > (Circle circle1, Circle circle2) {

return (circle1.compareTo(circle2) > 0);

}

friend bool operator >= (Circle circle1, Circle circle2) {

return (circle1.compareTo(circle2) >= 0);

}

friend bool operator == (Circle circle1, Circle circle2) {

return (circle1.compareTo(circle2) == 0);

}

friend bool operator != (Circle circle1, Circle circle2) {

return (circle1.compareTo(circle2) != 0);

}

};

int main() {

Circle circle1(5.0);

Circle circle2(7.0);

cout << "The area of the circle with radius of: " << circle1.getR() << " is " << circle1.getArea() << endl;

cout << "The area of the circle with radius of: " << circle2.getR() << " is " << circle2.getArea() << endl;

cout << "circle1 " << "> " << "circle2 is: " << ((circle1 > circle2) ? "true" : "false") << endl;

cout << "circle1 " << ">= " << "circle2 is: " << ((circle1 >= circle2) ? "true" : "false") << endl;

cout << "circle1 " << "<= " << "circle2 is: " << ((circle1 <= circle2) ? "true" : "false") << endl;

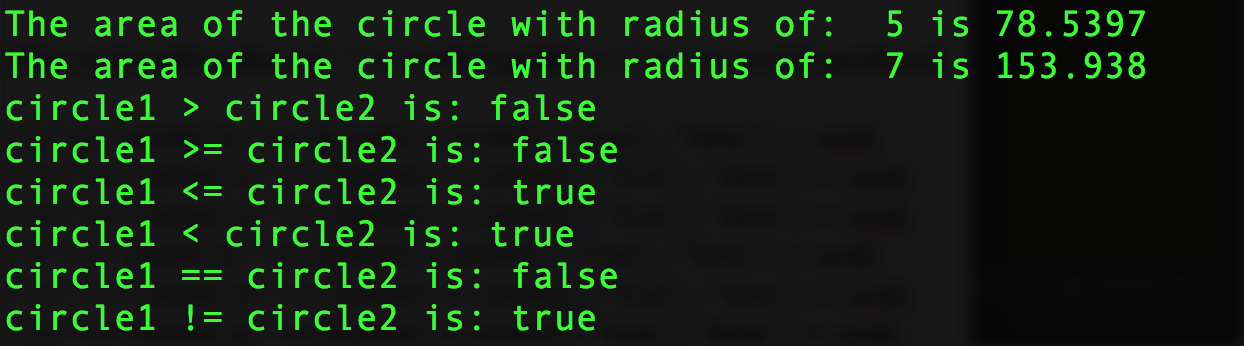
cout << "circle1 " << "< " << "circle2 is: " << ((circle1 < circle2) ? "true" : "false") << endl;

cout << "circle1 " << "== " << "circle2 is: " << ((circle1 == circle2) ? "true" : "false") << endl;

cout << "circle1 " << "!= " << "circle2 is: " << ((circle1 != circle2) ? "true" : "false") << endl;

return 0;

}



#include <iostream>

#include <string>

using namespace std;

class Person

{

protected:

string name, address, phoneNumber, email;

public:

virtual string toString() const

{

return "Person";

}

};

class Student1 : public Person

{

private:

int status;

public:

string toString() const

{

return "Student";

}

};

class MyDate

{

public:

int day, month, year;

};

class Employee : public Person

{

protected:

string office;

int salary;

MyDate dateHired;

public:

string toString() const

{

return "Employee";

}

};

class Faculty : public Employee

{

private:

string officeHours;

int rank;

public:

string toString() const

{

return "Faculty";

}

};

int main()

{

Faculty f;

Person p;

Employee e;

cout << f.toString() << endl;

cout << p.toString() << endl;

cout << e.toString() << endl;

return 0;

}

