Варіант 2

Структура

struct MyFrac

{

public long nom, denom;

public MyFrac(long nom\_, long denom\_)

{

// вміст допишіть самостійно, ВРАХОВУЮЧИ ВСІ ВКАЗІВКИ

}

public override string ToString()

{

// вміст допишіть самостійно; якщо врахувати всі вказівки

// щодо інших складових, саме тут усе виявиться ДУЖЕ просто

}

}

подає звичайний дріб у формі чисельника і знаменника.

Реалізувати функції (методи):

* конструктор MyFrac(long n, long d), який створює структуру за наданими чисельником і знаменником; ; цей конструктор, як і будь-який конструктор, має бути оголошений всередині структури, як вказано вище; конструктор *повинен* займатися скороченням (наприклад, перетворювати 15/40 у 3/8 — використайте алгоритм Евкліда знаходження НСД; його можна знайти у багатьох місцях, зокрема — розділ «Інваріант цикла» посібника з дискретної математики); також, треба забезпечити, щоб не був від’ємним (якщо передали від'ємний — змінити знак і чисельника, і знаменника);
* public override string ToString(), всередині структури, як вказано вище (див. також спільний текст обох варіантів) — формуватиме рядкове подання дробу, у вигляді, наприклад, «17/7» , або «-2/3»;

Всі подальші функції (методи) робити *не* всередині цієї структури, а в основному класі (який, якщо Ви нічого не змінювали, називається class Program). Це порушує класичний ООП-підхід, але не будемо знову повторювати все, вже сказане у спільній для обох варіантів частині:

* static string ToStringWithIntegerPart(MyFrac f) — формуватиме рядкове подання дробу з виділеною цілою частиною; наприклад, для того ж дробу 17/7 *ця* функція повинна вертати «(2+3/7)»; мінус може бути перед усіма дужками на позначення від'ємності дробу, ніякі інші мінуси не дозволені;
* static double DoubleValue(MyFrac f) — формуватиме дійсне значення дробу (можливо, приблизне за рахунок похибок);
* static MyFrac Plus(MyFrac f1, MyFrac f2) — рахуватиме суму двох дробів f1 та f2; при поверненні результата *обов'язково* використовувати конструктор, тоді про інші засоби скорочення дробу можна не дбати;
* static MyFrac Minus(MyFrac f1, MyFrac f2), яка рахує різницю двох дробів f1 та f2 — усі ті самі вимоги, що для plus;
* static MyFrac Multiply(MyFrac f1, MyFrac f2), яка рахує добуток двох дробів f1 та f2 — усі ті самі вимоги, що для plus;
* static MyFrac Divide(MyFrac f1, MyFrac f2), яка рахує частку двох дробів f1 та f2 — усі ті самі вимоги, що для plus; можливістю ділення на нуль знехтувати.
* static MyFrac GetRGR113LeftSum(int n) — рахуватиме (в типі MyFrac) суму 1/(1\*3)+1/(3\*5)+1/(5\*7)+...+1/((2n–1)\*(2n+1)). Переконайтеся, що ця функція вертає значення, рівні значенню виразу n/(2n+1). (При бажанні, можна довести, що така рівність правильна, методом математичної індукції; перш(ий/а) з усього курсу, хто це зробить, навіть отримає невеличкі бонусні бали в дискретну математику, але лише одна особа.)
* static MyFrac GetRGR115LeftSum(int n) — рахуватиме (в типі MyFrac) добуток (1–1/4)\*(1–1/9)\*(1–1/16)\*…\*(1–1/n2). Переконайтеся, що ця функція вертає значення, рівні значенню виразу (n+1)/(2n). (При бажанні, можна довести, що така рівність правильна, методом математичної індукції; перш(ий/а) з усього курсу, хто це зробить, навіть отримає невеличкі бонусні бали в дискретну математику, але лише одна особа. Причому, одна й та ж особа не може отримувати ці бонусні бали за обидва вирази.)

Функції (методи) GetRGR115LeftSum та GetRGR115LeftSum повинні виклик*а*ти відповідні арифметичні функції (методи) Plus, Minus, Multiply; всі арифметичні функції (методи) Plus, Minus, Multiply, Divide повинні виклик*а*ти конструктор; інших використань функцій зсер*е*дини інших функцій не вимагається, бо недоцільно.

Нарешті, рекомендованими є, наприклад, такі реалізація двох зі згаданих методів:

static MyFrac Plus(MyFrac f1, MyFrac f2)

{

return new MyFrac(f1.nom \* f2.denom + f1.denom \* f2.nom,

f1.denom \* f2.denom);

}

Тут використано відому формулу (*a*/*b*)+(*c*/*d*) = (*ad*+*bc*)/(*bd*), врахувавши, що те, що в класично-математичному записі цієї формули позначається *a*, *b*, *c*, *d*, є чисельником першого дробу, знаменником першого дробу, чисельником другого дробу та знаменником другого дробу відповідно, а отже, у позначеннях, які слідують із факту використання структур та вказаного заголовку функції (методу), це f1.nom, f1.denom, f2.nom та f1.nom відповідно.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Laba4

{

struct MyFrac

{

public long nom, denom;

public MyFrac(long nom\_, long denom\_)

{

long abs\_nom\_ = Math.Abs(nom\_);

long abs\_denom\_ = Math.Abs(denom\_);

nom = nom\_;

denom = denom\_;

while (abs\_nom\_ > 0 && abs\_denom\_ > 0)

{

if (abs\_nom\_ > abs\_denom\_)

abs\_nom\_ %= abs\_denom\_;

else abs\_denom\_ %= abs\_nom\_;

}

long res = abs\_nom\_ + abs\_denom\_;

if (nom\_ < 0 && denom\_ < 0)

{

res \*= -1;

}

nom /= res;

denom /= res;

}

public override string ToString()

{

string outputString = nom + "/" + denom;

return outputString;

}

}

class Program

{

static MyFrac Plus(MyFrac f1, MyFrac f2)

{

return new MyFrac(f1.nom \* f2.denom + f1.denom \* f2.nom, f1.denom \* f2.denom);

}

static MyFrac Minus(MyFrac f1, MyFrac f2)

{

return new MyFrac(f1.nom \* f2.denom - f1.denom \* f2.nom, f1.denom \* f2.denom);

}

static double DoubleValue(MyFrac f)

{

return (double)f.nom / f.denom;

}

static MyFrac Multiply(MyFrac f1, MyFrac f2)

{

return new MyFrac(f1.nom \* f2.nom, f1.denom \* f2.denom);

}

static MyFrac Divide(MyFrac f1, MyFrac f2)

{

return new MyFrac(f1.nom \* f2.denom, f1.denom \* f2.nom);

}

static MyFrac GetRGR113LeftSum(int n)

{

MyFrac res = new MyFrac(1, 1);

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

{

res = Plus(res, Divide(new MyFrac(1, 1), new MyFrac((2 \* i - 1) \* (2 \* i + 1), 1)));

}

return Minus(res, new MyFrac(1, 1));

}

static MyFrac GetRGR115LeftSum(int n)

{

MyFrac res = new MyFrac(1, 1);

for (int i = 2; i <= n; i++)

{

res = Multiply(res, Minus(new MyFrac(1, 1), new MyFrac(1, i \* i)));

}

return res;

}

static void Main(string[] args)

{

Console.WriteLine("Enter the numerator");

long nom\_ = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("\nEnter the denomerator");

long denom\_ = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("\nEnter a number");

MyFrac myFrac = new MyFrac(nom\_, denom\_);

MyFrac myFrac1 = new MyFrac(15, 40);

int n = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("\nFirst fraction = " + myFrac);

Console.WriteLine("\nSecond fraction = " + myFrac1);

Console.WriteLine("\nAddition of fractions\nResult : " + Plus(myFrac, myFrac1));

Console.WriteLine("\nSubtraction of fractions\nResult : " + Minus(myFrac, myFrac1));

Console.WriteLine("\nDouble value of fraction 1 is: {0}\nDouble value of fraction 1 is: {1}", DoubleValue(myFrac), DoubleValue(myFrac1));

Console.WriteLine("\nMultiplication of fractions\nResult : " + Multiply(myFrac, myFrac1));

Console.WriteLine("\nDivision of fractions\nResult : " + Divide(myFrac, myFrac1));

Console.WriteLine("\nGetRGR113LeftSum\nResult : " + GetRGR113LeftSum(n));

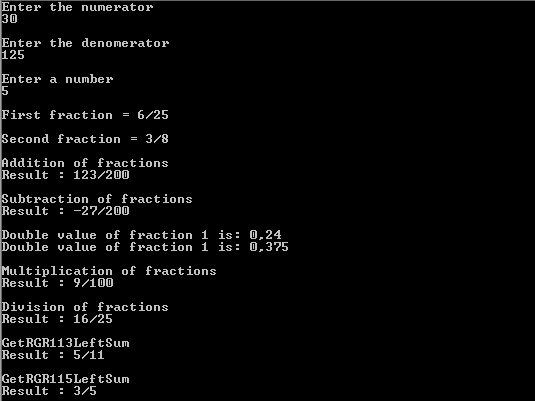
Console.WriteLine("\nGetRGR115LeftSum\nResult : " + GetRGR115LeftSum(n));

Console.ReadKey();

}

}

}



|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

