Ejercicio 1-2

Algoritmo de Euclides y sobrecarga del flujo de salida

Algoritmos y Estructuras de Datos

Tema 1: análisis de algoritmos no recursivos

1º Grado en Ingeniería en Desarrollo de Contenidos Digitales
Profesor Dr. Carlos Grima Izquierdo (www.carlosgrima.com)
U-tad (www.u-tad.com), curso 2013/14

Ampliar el ejercicio 1-1, implementando el método "simplificar()". Para ello, se implementará primero un método privado llamado "calcularMCD" que calcule el máximo común divisor de dos números naturales (no enteros).

- "calcularMCD" calculará el máximo común divisor de dos naturales pasados como parámetros, usando el algoritmo de Euclides. En realidad se pasarán dos "int", pero se añadirán las precondiciones de que tienen que ser mayores o iguales que cero.

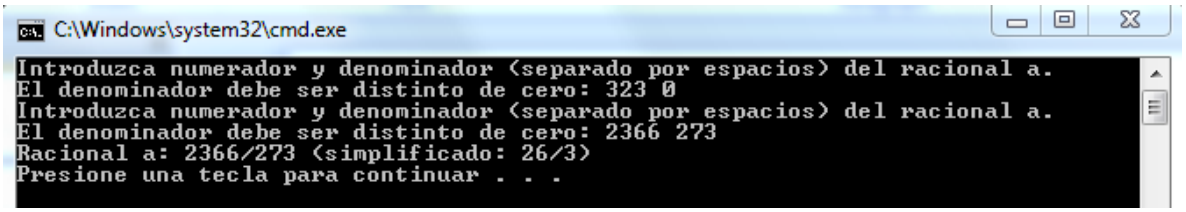
- "simplificar" averiguará el MCD del numerador y el denominador, y dividirá ambos entre dicho MCD.

Modificar también el ejercicio sustituyendo la función escribir() por una sobrecarga del operador “<<” para que reconozca un objeto de la clase Racional y por lo tanto lo pueda imprimir por pantalla usando “cout” con el mismo formato que el antiguo escribir.

Realizar un método main que pruebe nuestra clase:

1. Pedirá el numerador y el denominador de un racional ("a"). Si se introduce 0 como denominador, el programa seguirá pidiendo hasta que se meta un valor adecuado
2. Lo imprimirá por pantalla
3. Lo simplificaremos e imprimiremos el resultado por pantalla de nuevo

El pantallazo de la prueba básica es:



Algunas observaciones, pistas y ayudas sobre cómo se recargan operadores en C++:

- Para recordar cómo se sobrecargan operadores en general podemos consultar http://www.zator.com/Cpp/E4_9_18.htm (junto con todas sus subsecciones, que está en el menú de arriba a la izquierda) y/o http://c.conclase.net/curso/?cap=035#inicio (con todas sus subsecciones también).
- Ojo con cómo se pasan los parámetros y se devuelven los resultados en la sobrecarga de operadores y en los constructores de copia. En la mayoría de los casos hay que hacerlo por referencia, no por valor. Repasa la diferencia exacta entre punteros y referencias en C++.
- Para recordar cómo se sobrecargan los operadores de flujo "<<" y ">>", observa el siguiente ejemplo completo (todo ello en un archivo "main"):

```
#include <iostream>
using namespace std;

class Complejo {
	float x, y;
public:
	Complejo (Complejo& c) {x=c.x; y=c.y; cout << "Constructor copia\n";}
	Complejo (float x, float y) {
		this->x = x; // Hay que usar "this" para distinguir
		this->y = y;
		cout << "Constructor\n";
	}
	friend ostream& operator << (ostream& salida, const Complejo &c);
	friend istream& operator >> (istream& entrada, Complejo &c);
};

void main () {
	Complejo v1(1,0);
	cout << "Complejo sin actualizar: " << v1 << endl;
	cout << "Introduzca nuevos valores x e y separados por espacio: ";
	cin >> v1;
	cout << "Complejo actualizado: " << v1 << endl;
}

ostream& operator<<(ostream &salida,const Complejo &p)
{
	salida << "("<<p.x<<","<<p.y<<")";
	return salida;
}

// Entramos las dos coordenadas separadas por espacio
istream& operator>>(istream &entrada,Complejo &p)
{
	entrada >> p.x >> p.y;
	return entrada;
}
```