

Programación 4

PRÁCTICO 1

Los ejercicios marcados con * se consideran indispensables para adquirir los conocimientos básicos del curso.

Ejercicio 1 *

Exponer la metodología por usted utilizada hasta el momento para el desarrollo de software. Identificar los pasos seguidos en el proceso de desarrollo desde la especificación del problema hasta la obtención del programa; ¿qué herramientas utiliza para realizar una especificación formal de la realidad?, ¿y para especificar el diseño de la solución?, ¿y para realizar el pasaje de una a otra? ¿Con qué herramientas cuenta para gestionar la mantenibilidad, reusabilidad y evolutividad, cualidades deseables en los sistemas de software?

Ejercicio 2

Identificar objetos y links en la siguiente realidad sobre Pepe y su vida cotidiana. ¿Qué abstracciones pueden realizarse para dicha realidad?

Debido al aumento del crudo y la consecuente suba del boleto montevideano, Pepe ha tenido que tomar la decisión de conducir su bicicleta para ir a estudiar a la Facultad de Ingeniería.

Ejercicio 3

Recientemente los restaurantes han implementado un servicio de atención al cliente directamente en su vehículo. El mecanismo de funcionamiento de este servicio es el siguiente.

El cliente se aproxima a la ventanilla y le dice al recepcionista su pedido. El recepcionista le indica al cocinero que prepare la comida solicitada. Le indica al cajero que prepare la cuenta y le pide al cliente que pase a la siguiente ventanilla. El cliente llega a la segunda ventanilla y le paga al cajero, quién, luego de cobrar, le indica que siga a la siguiente ventanilla en donde el cocinero le entrega el pedido.

Modelar la realidad como una interacción de objetos. Abstractar los conceptos presentes en dicha realidad.

Ejercicio 4 *

Implementar en C++ una clase que represente un punto en un plano utilizando coordenadas cartesianas. Proveer operaciones para obtener y modificar la posición del punto en el plano.

A su vez, implementar en C++ una clase que represente un segmento en un plano. Proveer una operación que calcule el largo del segmento.

Ejercicio 5 *

- a) Implementar en C++ una clase llamada Fecha que permita definir y manipular fechas, incluyendo datos de día, mes y año. La clase deberá proveer las siguientes operaciones.

```
avanzar : Fecha x Entero -> Fecha
// Devuelve una fecha resultante de sumar una cantidad dada de días
```

```
retroceder : Fecha x Entero -> Fecha
// Devuelve una fecha resultante de restar una cantidad dada de días
```

```

diferencia : Fecha x Fecha -> Entero
// Devuelve la cantidad de días entre dos fechas
comparación : Fecha x Fecha -> Boolean
// Compara dos fechas (==, !=, <, >, <=, >=)

```

- b) Agregar constructores, destructor y operador de asignación, según considere necesario.
- c) Agregar la sobrecarga del operador de inserción de flujo (<<) en un objeto de tipo `std::ostream`. El mismo debe imprimir en el flujo de salida la fecha en el siguiente formato: AAAA/MM/DD.

Ejercicio 6 *

- a) Implementar en C++ el datatype `String`, que represente cadenas de caracteres de largo variable. Debe ser posible obtener un `String` tanto a partir de una cadena de caracteres (`char *`) como a partir de otro `String`.
- b) Agregar la sobrecarga de los siguientes operadores:
 - Asignación.
 - Concatenación (operador `+`).
 - Comparación.
 - Acceso a un caracter del `String` (operador `[]`).
- c) Agregar la sobrecarga de los operadores de inserción y extracción de flujo.
- d) Agregar las siguientes operaciones:
 - `largo`, que retorna la cantidad de caracteres de la cadena.
 - `substring`, que retorna el substring que se encuentra entre dos posiciones dadas de la cadena (incluyendo los caracteres en dichas posiciones).
- e) Agregar constructores y destructor según considere necesario.
- f) Agregar manejo de excepciones de forma que:
 - Al intentar acceder a una posición del `String` inválida se lance la excepción `"std::out_of_range"`.
 - Al recibir un parámetro inválido en una operación se lance la excepción `"std::invalid_argument"`.

Ejercicio 7

- a) Implementar en C++ una clase llamada `Racional` que permita definir y manipular números racionales, representados como un par de enteros. La clase deberá proveer las siguientes operaciones.

```

suma          : Racional x Racional -> Racional
diferencia    : Racional x Racional -> Racional
producto      : Racional x Racional -> Racional
cociente      : Racional x Racional -> Racional
igualdad      : Racional x Racional -> bool

```

- b) Agregar a la implementación de la clase `Racional` las operaciones habituales entre racionales y enteros. Sobrecargar los operadores que se consideren necesarios. Como ejemplo, la clase `Racional` debería soportar la siguiente aplicación:

```

void main() {
    Racional r1(2,3), r2;
    Racional r3(4), r4 = r1;

    if (r1 == r2)
        r1 = r1 + r2;
}

```

```

else
    r1 = r1 - r2;

if (r3 != r4)
    r3 = r3 * r4;
else
    r3 = r3 / r4;

if (r1 == 2)
    r1 = r1 + 1;
else
    r1 = r1 - 1;

if (r3 != 3)
    r3 = r3 * 2;
else
    r3 = r3 / 2;
}

```

- c) ¿Qué modificaciones habría que realizarle a la implementación para que soporte además la siguiente aplicación?

```

r3 = 3 * r2;
if (4 == (2 + r2))
    r3 = 1 / r2;

```

Ejercicio 8 *

Considerar definidas las clases A, B y C, las asociaciones R1 entre las clases A y B, y R2 entre las clases B y C. Sean los objetos a, b y c, de clases A, B y C respectivamente. ¿Es posible que existan los siguientes links?

- a) Un link entre a y b
- b) Un link entre b y c
- c) Un link entre a y c
- d) Dos links entre b y c

Ejercicio 9 *

Considerar definidas las clases A, B, C, D y E, que cumplen $B <: A$, $C <: A$, $D <: B$ y $E <: D$.

- a) Indicar:
 - i) Clase base directa de B
 - ii) Clase base directa de E
 - iii) Subclases indirectas de A
 - iv) Subclases directas de B
- b) Definir la propiedad subsumption y explicar informalmente su significado.
- c) Considerando las propiedades de cada clase (atributos, asociaciones, operaciones): ¿Qué propiedades tiene la clase A?, ¿Qué propiedades tiene la clase E?

Ejercicio 10

Considerar definidas las clases A, B, C, D y E, que cumplen $B <: A$, $C <: A$, $D <: B$ y $E <: D$.

Suponer que A tiene una operación polimórfica $g()$ que se encuentra redefinida en el resto de las clases menos en D y tiene otra operación $h()$ no polimórfica. Discutir qué método se ejecuta dependiendo de la instancia, tipo de despacho y operación invocada.

Ejercicio 11

Considerar definidas las clases D, E y F, que cumplen $F <: E$, y una operación global con la siguiente signatura.

```
void g(E param)
```

Considerar además que se cuenta con los objetos d, e y f, instancias directas de las clases D, E y F respectivamente.

Indicar cuáles de las siguientes invocaciones son válidas. Justificar.

- a) $g(d)$
- b) $g(e)$
- c) $g(f)$

Ejercicio 12 *

- a) Identificar clases, atributos, operaciones y métodos en la siguiente descripción del diseño de una parte de un sistema de gestión de personal.

Se considera una empresa que mantiene información de sus empleados. De cada empleado se almacena su nombre y se cuenta con una operación para obtener el mismo. Los empleados de la empresa pueden ser comunes, de los cuales se conoce su sueldo, o jornaleros, de los cuales se conoce la cantidad de horas que trabajó y el valor de la hora. Todos los meses, al realizarse la liquidación de los sueldos, la empresa calcula el monto total por concepto de sueldos, en base al sueldo fijo para empleados comunes y como la cantidad de horas por el valor de la hora para empleados jornaleros.

- b) ¿Existe alguna clase abstracta en su solución? ¿En qué situaciones debemos definir a una clase como abstracta?
- c) Dar una descripción (pseudocódigo) de la operación `getTotal()` mediante la cual la empresa calcula el monto total de la liquidación de todos sus empleados. Tener en cuenta que en un futuro pueden incorporarse nuevos tipos de empleados, y que la repercusión en la implementación existente debe ser mínima (en particular sobre las operaciones ya implementadas en clases existentes).
- d) Definir el concepto de polimorfismo e identificar su aplicación en el modelo realizado en la parte (b). ¿Qué beneficios le aportó el polimorfismo y cuál otra propiedad le fue de utilidad?
- e) Implementar en C++ el modelo anterior incluyendo la funcionalidad de obtener el total de sueldos.
- f) Realizar un procedimiento `main()` que permita, mediante un menú, ingresar una lista de empleados y que calcule el total de sueldos a pagar a los empleados de la empresa.

Ejercicio 13

- a) Implementar en C++ una clase que represente a los conjuntos de enteros, utilizando como estructura de datos arreglos dinámicos. La clase deberá proveer al menos las siguientes operaciones:

```

agregar      : SetInt x int -> SetInt
remover      : SetInt x int -> SetInt
union        : SetInt x SetInt -> SetInt
diferencia   : SetInt x SetInt -> SetInt
interseccion : SetInt x SetInt -> SetInt
pertenece    : SetInt x int -> bool
esVacio      : SetInt -> bool
cantidadElem : SetInt -> int
esIgual      : SetInt x SetInt -> bool

```

- b) Sobrecargar los operadores +, -, == y != para que realicen la unión, diferencia, igualdad y desigualdad de conjuntos.
- c) Ejecutar manualmente y en una computadora el siguiente programa comparando los resultados.

```

SetInt a,b,c,d,e;

a.agregar(0);
a.agregar(1);

b.agregar(1);
b.agregar(2);

c.agregar(0);
c.agregar(1);
c.agregar(2);

d = a + b;
e = c - b;

if (a == d)
    cout << "a == d\n";
else
    cout << "a != d\n";

if (e == a)
    cout << "e == a\n";
else
    cout << "e != a\n";

```

Ejercicio 14

- a) Implementar en C++, utilizando la estructura de datos lista enlazada, una clase llamada SecInt que permita definir y manipular secuencias de enteros. La clase deberá proveer las siguientes operaciones. Sobrecargar los operadores que se consideren necesarios.

```

largo          : SecInt -> int
insertarEnPosicion : SecInt x int x int -> SecInt
eliminarPosicion  : SecInt x int -> SecInt
obtenerPosicion   : SecInt x int -> int
contiene          : SecInt x int -> bool
concatenar        : SecInt x SecInt -> SecInt
igualdad          : SecInt x SecInt -> bool

```

- b) Ejecutar manualmente y en una computadora el siguiente programa comparando los resultados.

```

void main() {
    SecInt s1, s2;
    int pos = s1.largo();
}

```

```

s1.insertarEnPosicion(pos, 3);
s2.insertarEnPosicion(pos, 4);

SecInt s3 = s2;

s1.insertarEnPosicion(pos+1, s2.obtenerPosicion(pos));

if (s1.contiene(4))
    s2 = s3 + s1;

if (s3 == s2)
    for (int i = s3.largo()-1; i>=0; i--)
        s3.eliminarPosicion(i);
}

```

Ejercicio 15

- a) Implementar en C++ una clase llamada `MatrizInt` que represente matrices de enteros de tamaño fijo. La clase deberá proveer al menos las siguientes funcionalidades.

```

setPosicion      : MatrizInt x int x int x int -> MatrizInt
getPosicion      : MatrizInt x int x int -> int
getNumFilas      : MatrizInt -> int
getNumColumnas   : MatrizInt -> int
transpuesta      : MatrizInt -> MatrizInt

```

- b) Implementar un procedimiento `main()` que permita ingresar una matriz, calcule su transpuesta y la imprima en pantalla.