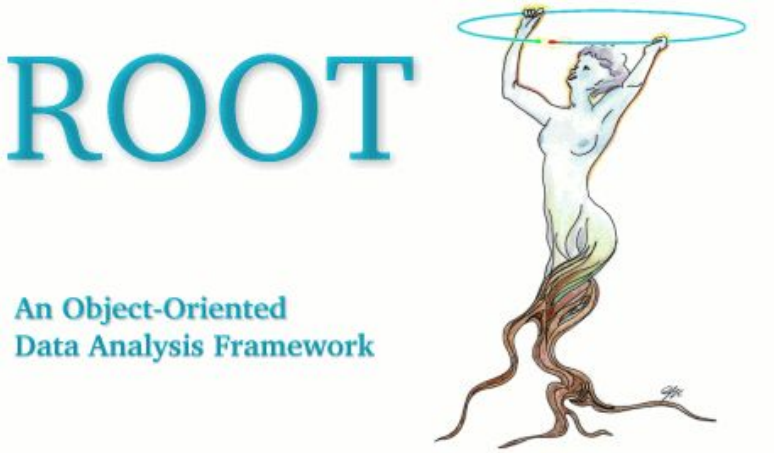
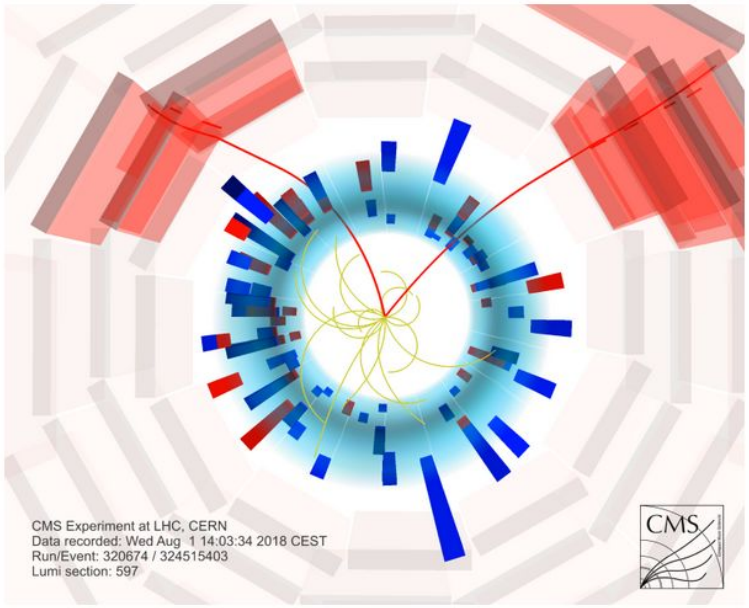
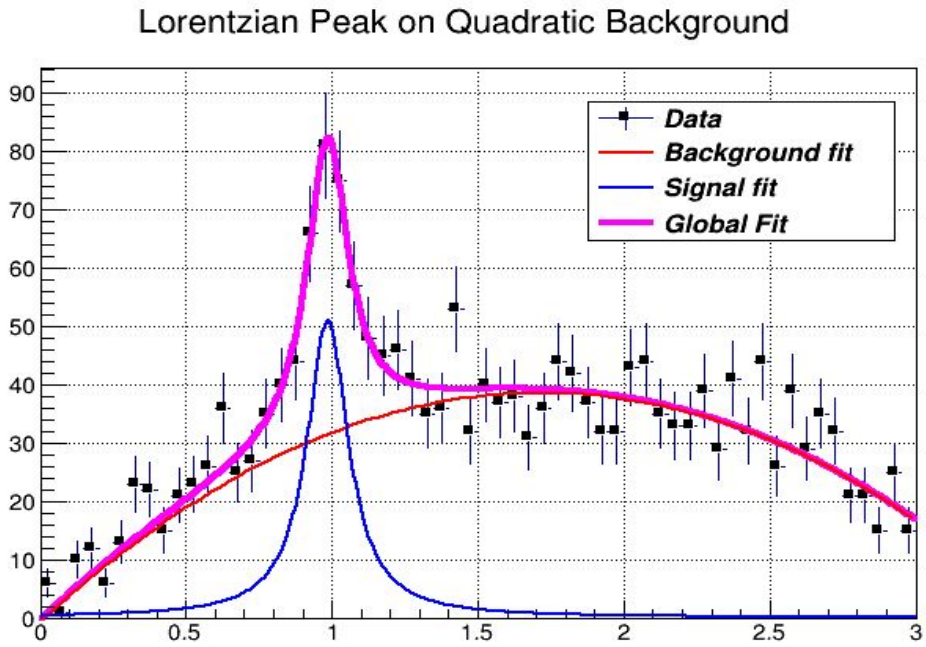






Big data With ROOT CERN

Jhovanny Andres Mejia Guisao

Centro Interdisciplinario de Investigación
y Enseñanza de la Ciencia

Motivation

What we hope to discuss about scientific data analysis?

- Advanced graphical user interface
- Interpreter for the C++ programming language
- Persistency mechanism for C++ objects
- Used to write every year petabytes of data recorded by the Large Hadron Collider experiments

Input and plotting of data from measurements and fitting of analytical functions.

New and Deleted: Administración dinámica de memoria

```
Tiempo *tiempoPtr;
tiempoPtr = new Tiempo;
delete tiempoPtr;
```

Si no se libera la memoria asignada en forma dinámica cuando ya no es necesaria, el sistema se puede quedar sin memoria antes de tiempo.
A esto se le conoce algunas veces como "**fuga de memoria**".

```
double *ptr = new double( 3.14159 );
Tiempo *tiempoPtr = new Tiempo( 12, 45, 0 );
```

```
int *arregloCalificaciones = new int[ 10 ];
delete [ ] arregloCalificaciones;
```

E. Codigos MorePointers

Task

in the lecture hour

Hacer una función que reciba una función y calcule la integral con las sumas de Riemann

For home

Calcular la norma de un vector

Orientación a objetos (POO). Object-oriented programming (OOP)

La orientación a objetos es un paradigma de programación que se centra en la creación de objetos y la manipulación de esos objetos para realizar una tarea o resolver un problema. Se basa en el concepto de "clases", que son plantillas para crear objetos, y en la "herencia", que permite crear nuevas clases a partir de clases existentes y así reutilizar el código.

La orientación a objetos también incluye otros conceptos clave como la encapsulación, que se refiere a la ocultación de la complejidad interna de un objeto y la exposición solo de una interfaz sencilla para interactuar con él; el polimorfismo, que permite que diferentes objetos respondan de manera diferente a un mismo mensaje o método; y la abstracción, que se refiere a la capacidad de simplificar y generalizar conceptos complejos para que puedan ser entendidos y utilizados más fácilmente en la programación.

En resumen, la orientación a objetos es una forma de organizar y estructurar el código de una aplicación o programa, basada en la creación y manipulación de objetos que interactúan entre sí para resolver un problema o realizar una tarea específica.

Orientación a objetos (POO). Object-oriented programming (OOP)

- **Objetos:**
  - **Es una variable creada a partir de una clase (instancia)**
  - **Es una variable que permite acceder a los datos y las funciones(métodos) de la clase**
- **Clase:**
  - **Es una plantilla que describe el objeto**
  - **Contiene variables que son llamadas atributos**
  - **Contiene funciones que son llamadas métodos**
- **Encapsulación**
  - **Permite cambiar la visibilidad atributos y métodos para con las palabras claves public, protected y private**
- **Atributos**
  - **Variables de la clase**
- **Métodos**
  - **Funciones de la clase**
- **Constructor**
  - **Es un método especial con el que se inicializa la clase.**
  - **No retorna valor**
  - **Los parámetros con con los que se instancian los objetos**
- **Destructor**
  - **Es un método especial para finalizar “limpiar” la clase.**
  - **No retorna valor**
  - **Se llama automáticamente**

Herencia

Si una clase derivada pudiera acceder a los miembros private de su clase base, las clases que heredan de esa clase derivada podrían acceder a esos datos también. Esto propagaría el acceso a lo que deben ser datos private , y se perderían los beneficios del ocultamiento de la información.

Clases base y clases derivadas

Clase base	Clases derivadas
Estudiante	EstudianteGraduado, EstudianteNoGraduado
Figura	Circulo, Triangulo, Rectangulo, Esfera, Cubo
Prestamo	PrestamoAuto, PrestamoMejoraHogar, PrestamoHipotecario
Empleado	Docente, Administrativo
CuentaBanco	CuentaCheques, CuentaAhorros

"**No digas que conoces a otro por completo, sino hasta que hayas dividido una herencia con él.**"
---------Johann Kasper Lavater

Herencia

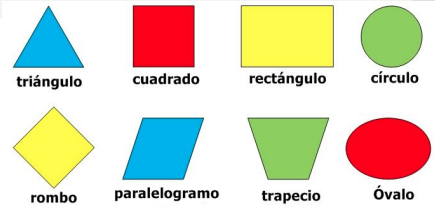
La herencia es un mecanismo fundamental en la programación orientada a objetos que permite crear una nueva clase a partir de una clase existente, llamada clase base o superclase. La clase creada a partir de la superclase se denomina clase derivada o subclase, y hereda todas las propiedades y métodos de la clase base, además de poder añadir nuevos miembros o métodos específicos.

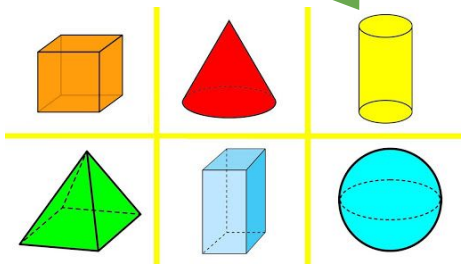
En la herencia, la clase base se utiliza como plantilla para la creación de la subclase, y ésta puede añadir nuevas propiedades o métodos, o bien modificar o sobrescribir los existentes. Además, la subclase también puede heredar de su superclase las propiedades y métodos públicos y protegidos, que se pueden utilizar en la subclase de la misma manera que en la superclase.

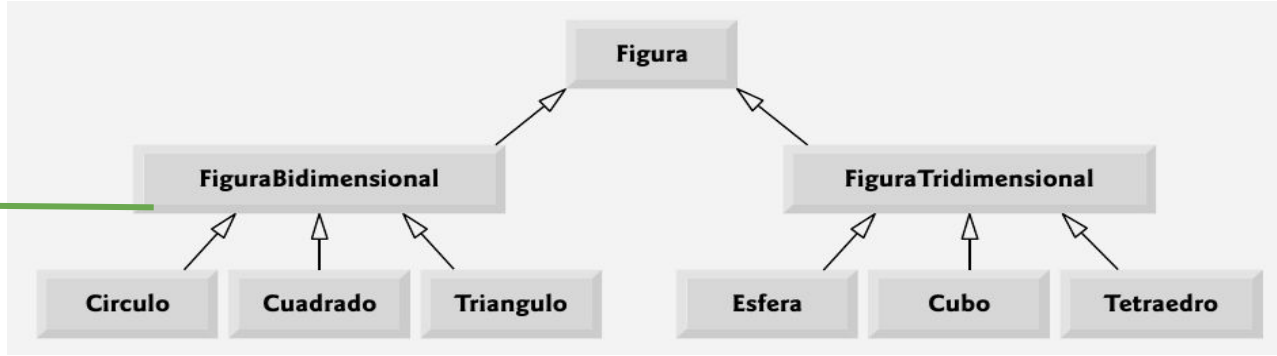
La herencia es un concepto importante en la programación orientada a objetos, ya que permite la creación de clases con características comunes que se pueden reutilizar en diferentes contextos, evitando la duplicación de código y mejorando la modularidad y la flexibilidad del diseño.

Clases base y clases derivadas

Recuerde: herencia significa "*es un*", composición significa "*tiene un*".







En lo que sigue por favor refiérase al archivo "plan.txt"

example1_clasebase1
example1_clasebase2

Copiar y pegar código de una clase a otra puede esparcir errores a través de varios archivos de código fuente. Para evitar duplicar código (y posiblemente los errores), use la herencia en vez de la metodología de “copiar y pegar” en situaciones en las que una clase debe “absorber” los datos miembro y las funciones miembro de otra clase.
Con la herencia, los datos miembro y las funciones miembro comunes de todas las clases en la jerarquía se declaran en una clase base. Cuando se requieren cambios para esas características comunes, es necesario realizar los cambios sólo en la clase base; así, las clases derivadas pueden heredar los cambios. **Sin la herencia, los cambios tendrían que realizarse en todos los archivos de código fuente que contengan una copia del código en cuestión.**

CMS-NOTE2023-007

Una mirada breve a la sobrecarga de operadores

Los operadores sobrecargados deben imitar la funcionalidad de sus contrapartes integrados; por ejemplo, el operador + debe sobrecargarse para realizar la suma, no la resta. Evite el uso excesivo o inconsistente de la sobrecarga de operadores, ya que esto puede ocasionar que el programa se haga críptico y difícil de leer.

Para sobrecargar un operador, **se escribe la definición de una función miembro no static** o la definición de una función global como se hace normalmente, excepto que el nombre de la función se convierte ahora en la **palabra clave operator** , seguida del símbolo del operador que se va a sobrecargar.

Operadores que no se pueden sobrecargar			
.	.*	::	?:

la "aridad" de un operador es el número de operandos que recibe. Tratar de cambiar la "aridad" de un operador a través de la sobrecarga de operadores es un error de compilación.

Tratar de crear nuevos operadores a través de la sobrecarga de operadores es un error de sintaxis.

Por lo menos un argumento de una función de operador debe ser un objeto o referencia de un tipo definido por el usuario.

examples1-4 (lecture7)

Una mirada breve al POLIMORFISMO

Para que en un programa se produzca el polimorfismo se tiene que seguir los siguientes pasos:

1. Crear una jerarquía de clases en la que operaciones importantes serán declaradas como métodos virtuales, puros o no.
2. Manipular los objetos a través de punteros.

POLIMORFISMO

La palabra "polimorfismo" significa "la facultad de asumir muchas formas", refiriéndose a la facultad de llamar a muchas funciones diferentes en una sola sentencia.

Funciones Virtuales

examples5-6 (lecture7)

Una función virtual es un método de una clase base que puede ser redefinida en cada una de las clases derivadas, podrá utilizar los métodos redefinidos en esta clase derivada.

Una función virtual se define cuando la palabra "virtual" antes de la declaración del método en la clase base.