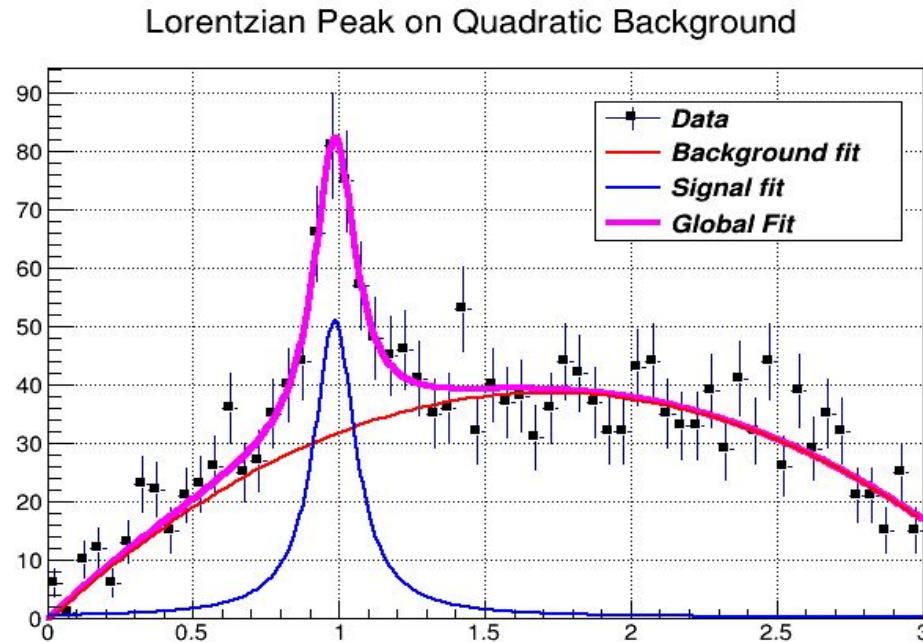
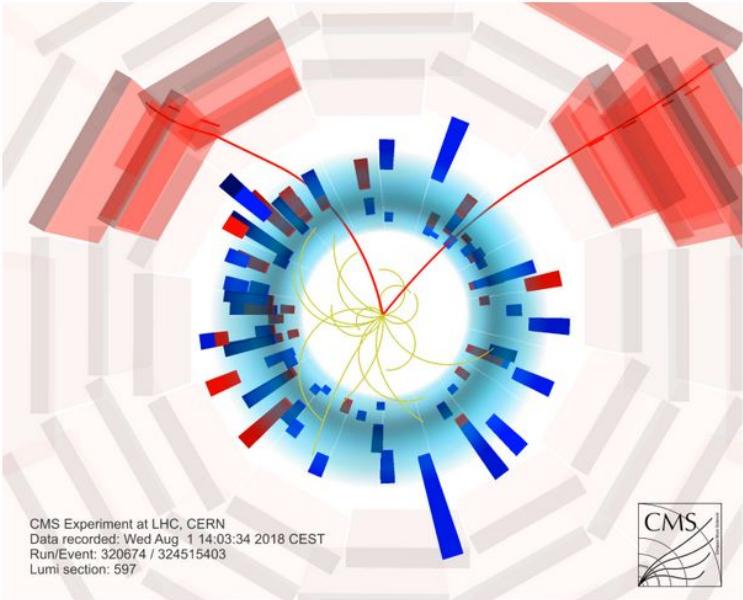


ROOT

An Object-Oriented
Data Analysis Framework



Big data With ROOT CERN

Jhovanny Andres Mejia Guisao
Centro Interdisciplinario de Investigación
y Enseñanza de la Ciencia

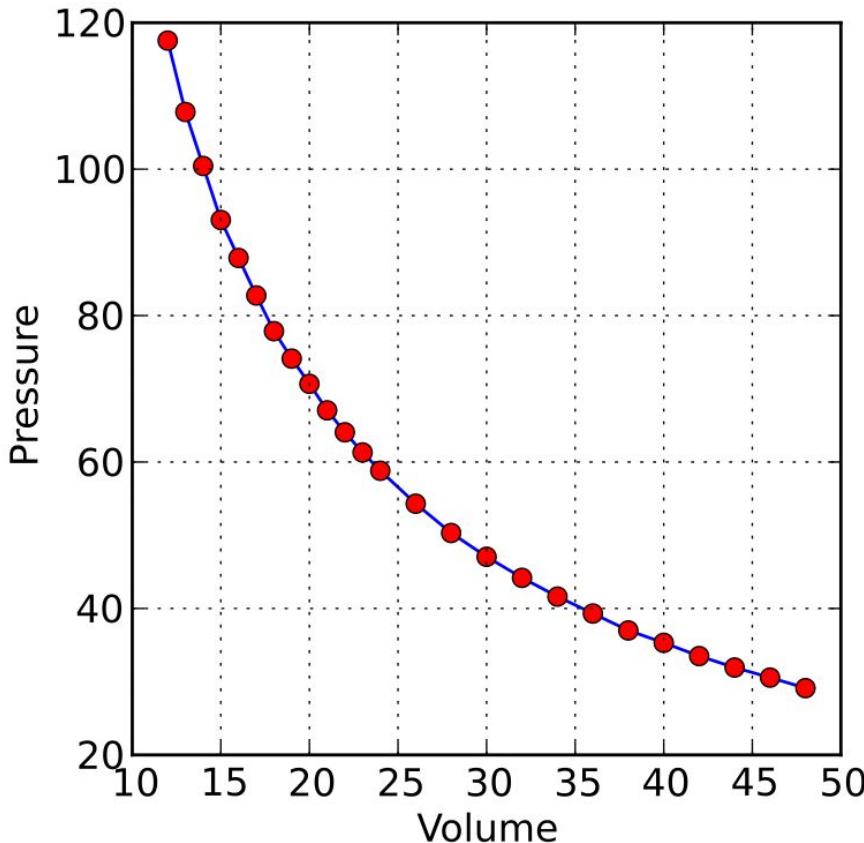
Motivation

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.

Motivation

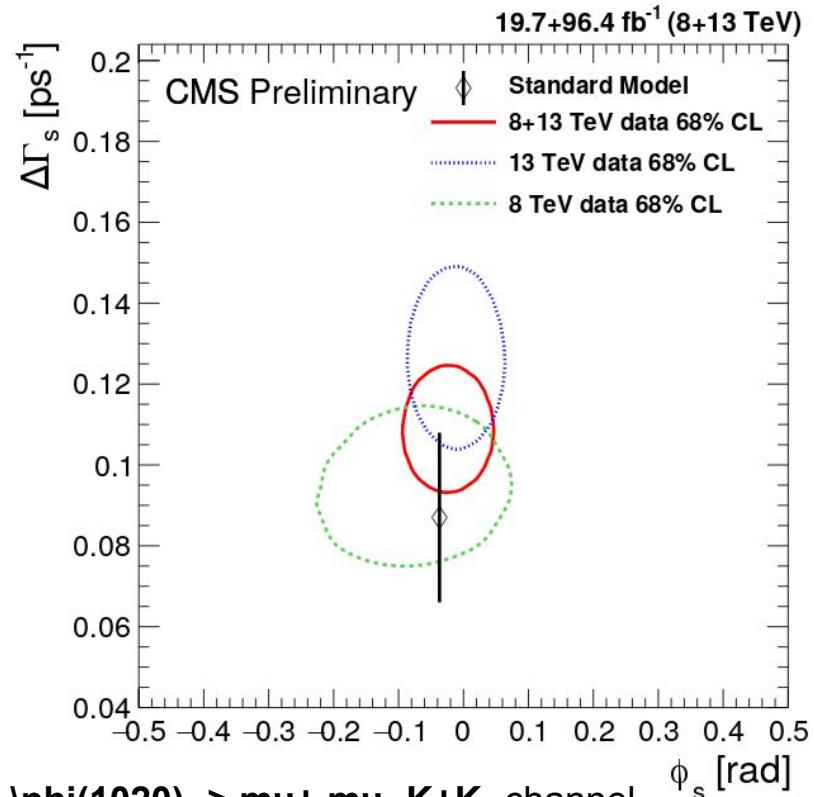
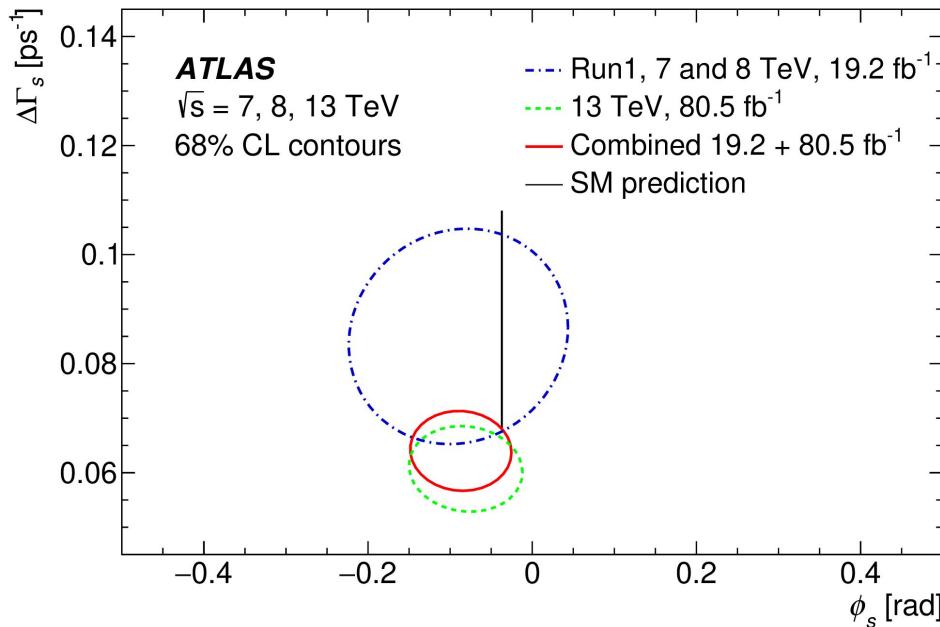


Para un gas a temperatura constante, el volumen es inversamente proporcional a la presión sobre éste [[link](#)]

Se puede explicar matemáticamente con:
 $pV = k$

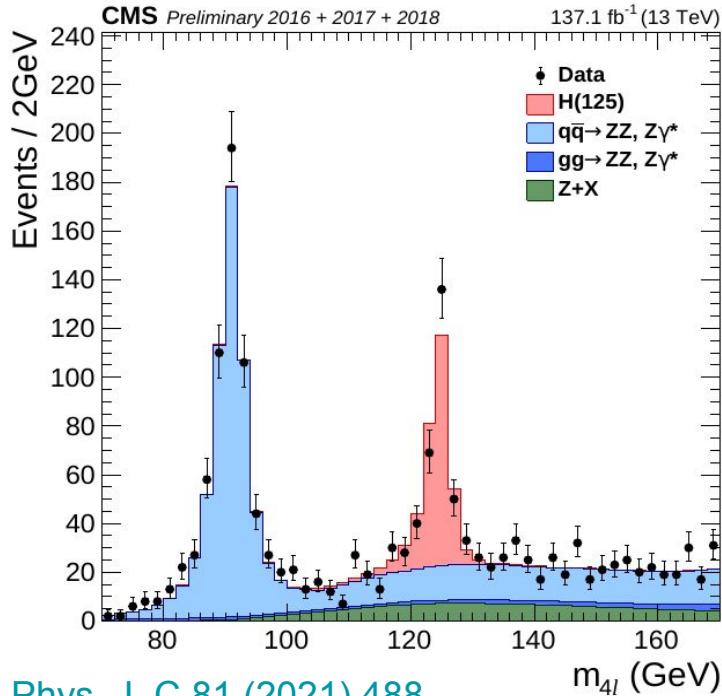
y la tarea del experimentador consiste en determinar la constante, k , a partir de un conjunto de medidas.

Motivation

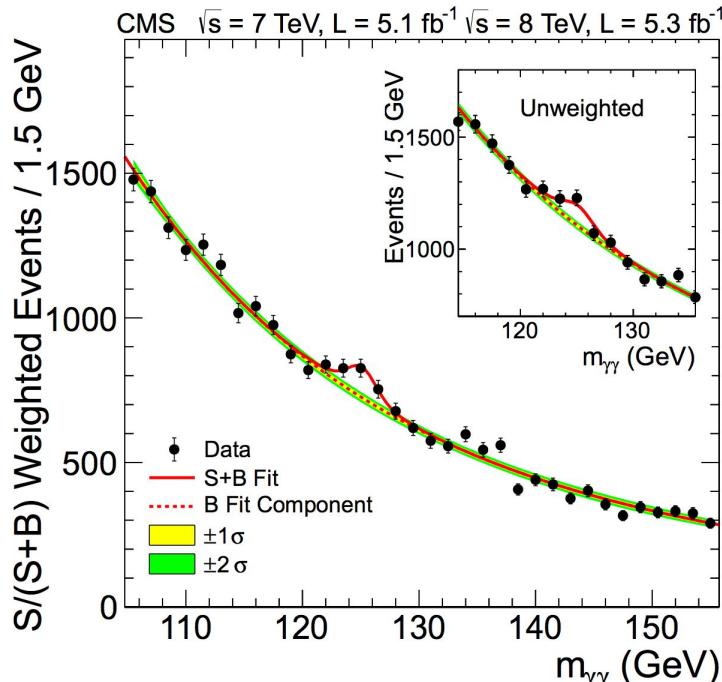


Measurement of the **CP violating phase** in the $B_s \rightarrow J/\psi \phi(1020) \rightarrow \mu^+ \mu^- K^+ K^-$ channel
in proton-proton collisions at $\sqrt{s} = 13 \text{ TeV}$ ([PLB 816 \(2021\) 136188](#))

Motivation



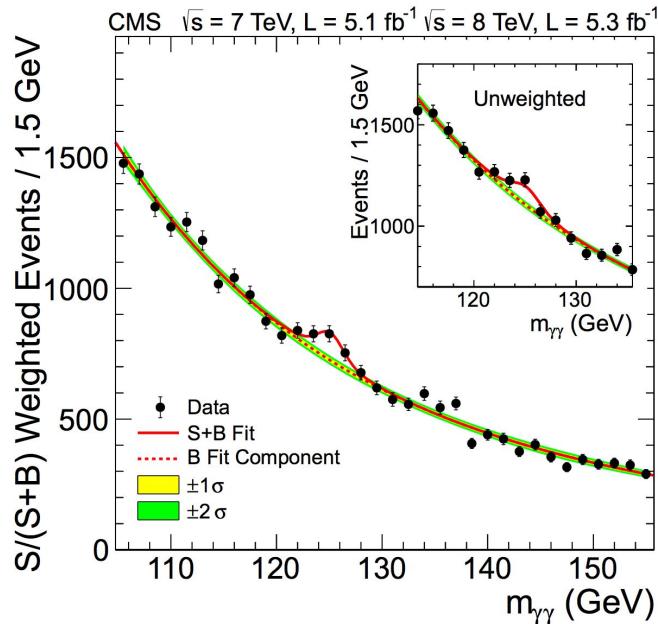
[Eur. Phys. J. C 81 \(2021\) 488](#)



Direct (left) and indirect (right) evidence for the Higgs boson. Left: the invariant mass of four leptons, showing evidence for the Higgs boson at $M_H \sim 126$ GeV. Right: the world average of the W boson mass vs the top-quark mass is shown. Overlayed are the contour plots when the direct Higgs boson mass measurement is used. It is evident that the indirect determination of the Higgs boson mass is quite consistent with the direct measurement.

Motivation

In Quantum mechanics, models typically only predict the **probability density function** (“pdf”) of measurements depending on a number of parameters, and the aim of the experimental analysis is to extract the parameters from the **observed distribution** of frequencies at which certain values of the measurement are observed. Measurements of this kind require means to generate and visualize frequency distributions, **so-called histograms**, and **stringent statistical treatment to extract the model parameters from purely statistical distributions**.



Motivation

Visualization of the data

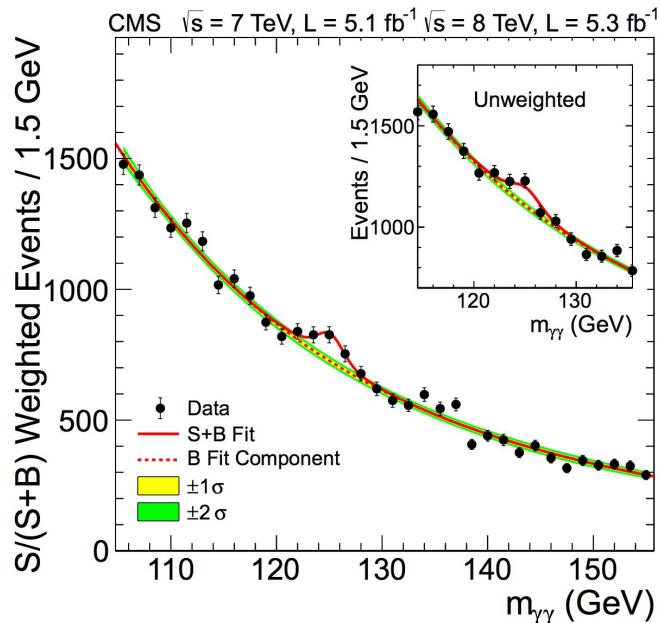
Corrections or parameter transformations?

One specialty of experimental physics are the inevitable uncertainties affecting each measurement.

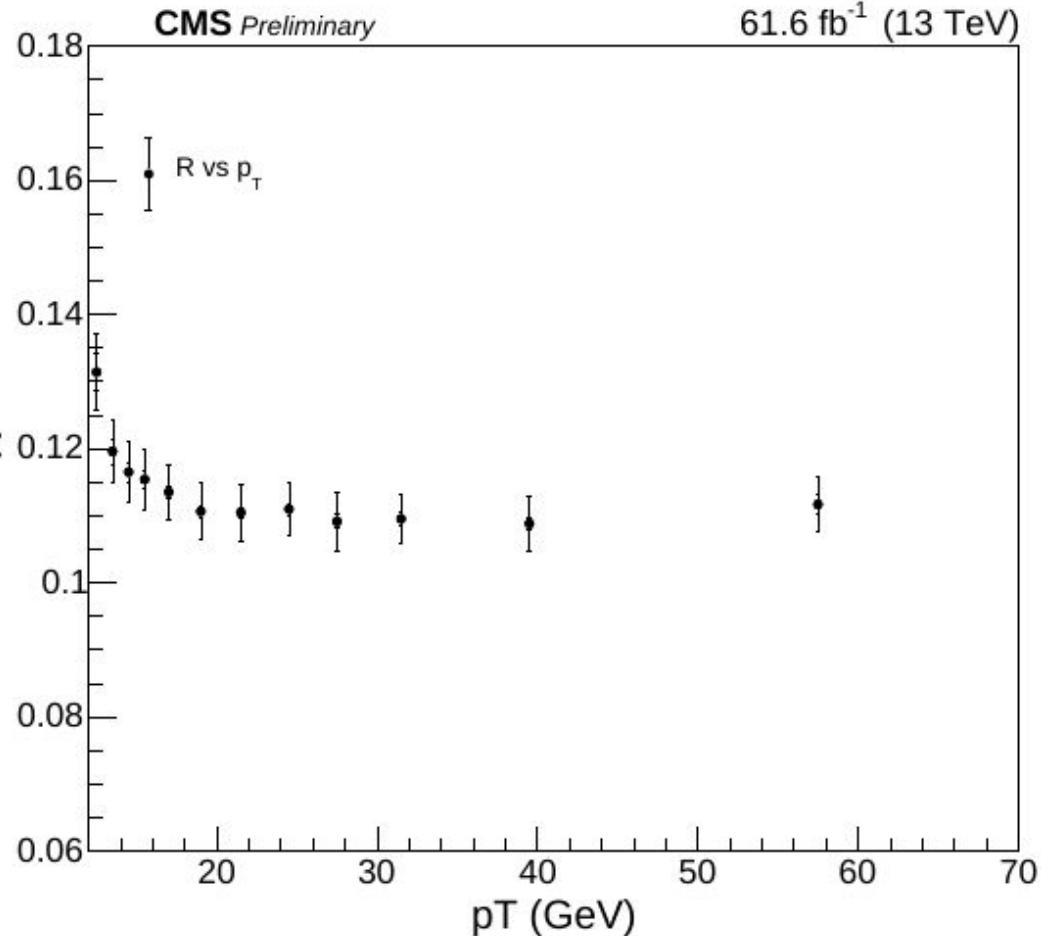
The statistical nature of the errors must be handled properly.

Quite often, the data volume to be analyzed is large - think of fine-granular measurements accumulated with the aid of computers. A usable tool therefore must contain easy-to-use and efficient methods for storing and handling data.

Simulation of expected data is another important aspect in data analysis. By repeated generation of “**pseudo-data**”, which are analysed in the same manner as intended for the real data, analysis procedures can be validated or compared.



Motivation



what does this distribution tell us?

why there are two errors? or more?

how can we understand/handle these errors?

[BPH-21-001](#)

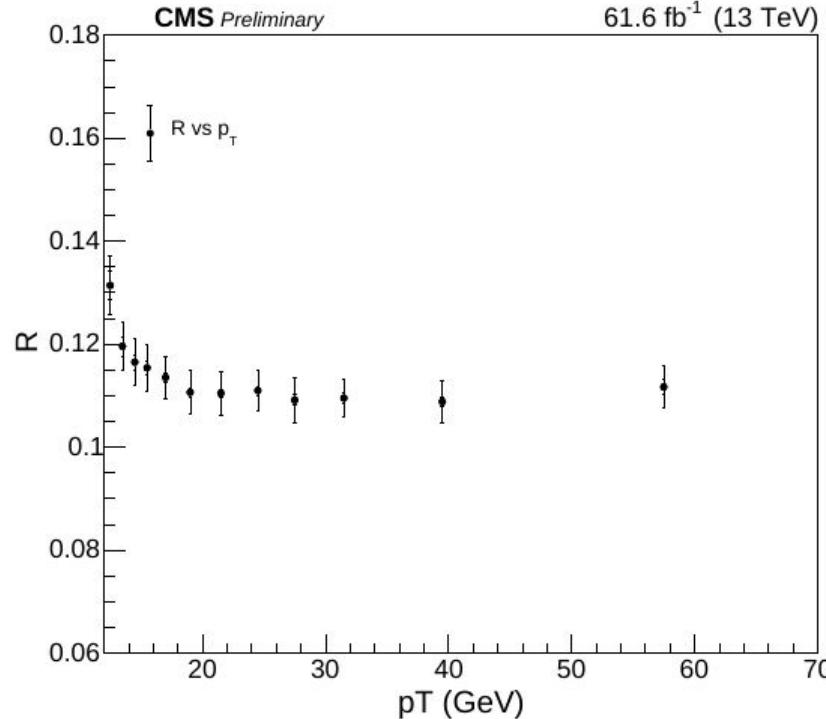
Motivation

Table 11: \mathcal{R} and uncertainty (σ_{tot}), including the statistical (σ_{stat}) uncertainty and the fully correlated and uncorrelated systematic uncertainties among the samples ($\sigma_{\text{sys}}^{\text{uncor}}$, $\sigma_{\text{sys}}^{\text{cor}}$). Correlations stem from the common tracking and fit model uncertainties.

p_T (GeV)	\mathcal{R}	σ_{tot}	σ_{stat}	$\sigma_{\text{sys}}^{\text{uncor}}$	$\sigma_{\text{sys}}^{\text{cor}}$
12 – 13	0.1314	0.0058	0.0028	0.0033	0.0038
13 – 14	0.1196	0.0046	0.0019	0.0015	0.0039
14 – 15	0.1165	0.0041	0.0015	0.0016	0.0035
15 – 16	0.1154	0.0042	0.0014	0.0018	0.0036
16 – 18	0.1135	0.0040	0.0009	0.0022	0.0032
18 – 20	0.1106	0.0041	0.0009	0.0022	0.0033
20 – 23	0.1105	0.0042	0.0008	0.0024	0.0034
23 – 26	0.1110	0.0040	0.0009	0.0023	0.0031
26 – 29	0.1091	0.0044	0.0010	0.0020	0.0038
29 – 34	0.1095	0.0037	0.0010	0.0022	0.0028
34 – 45	0.1088	0.0040	0.0009	0.0023	0.0032
45 – 70	0.1117	0.0041	0.0014	0.0021	0.0033

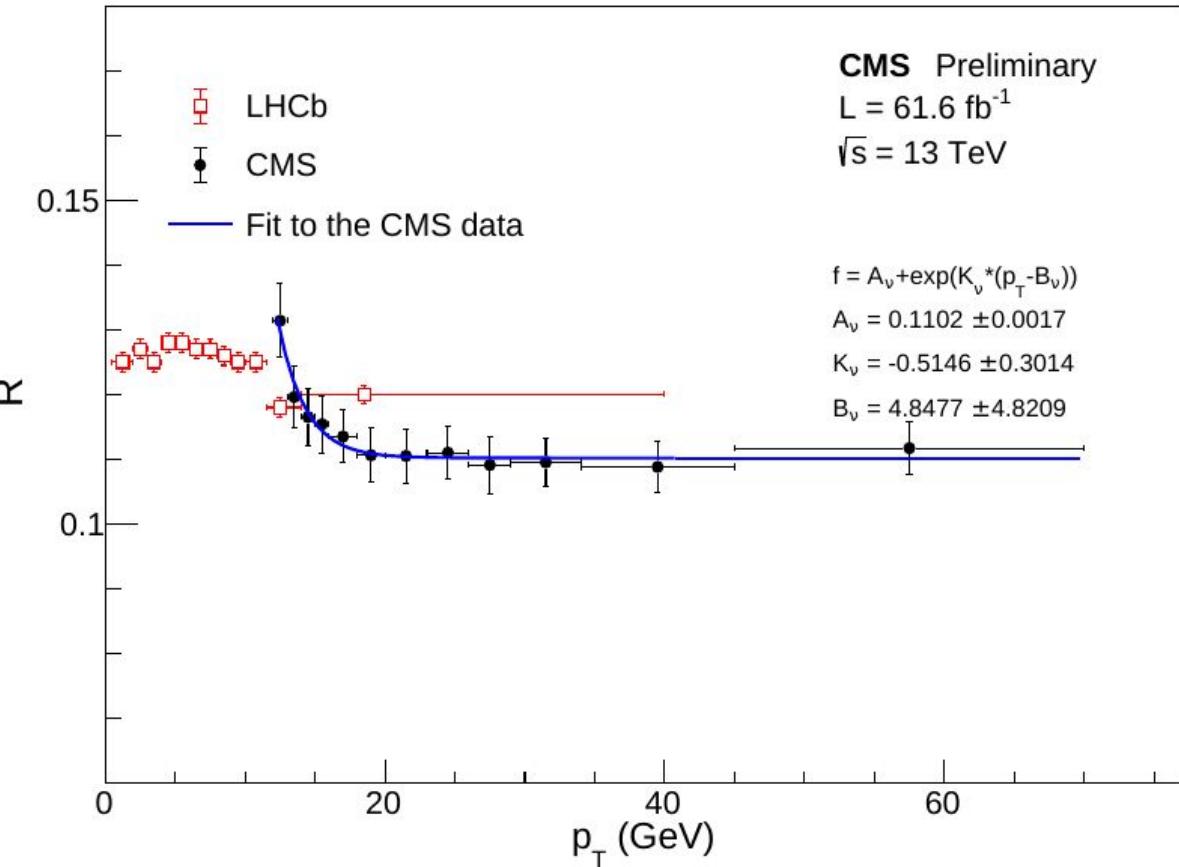
3.214	0.663	0.545	0.571	0.450	0.513	0.520	0.421	0.677	0.313	0.464	0.473
0.663	2.242	0.645	0.676	0.533	0.608	0.615	0.498	0.801	0.370	0.549	0.560
0.545	0.645	2.005	0.556	0.438	0.499	0.506	0.410	0.658	0.304	0.451	0.460
0.571	0.676	0.556	1.973	0.459	0.523	0.530	0.429	0.690	0.319	0.473	0.482
0.450	0.533	0.438	0.459	1.721	0.413	0.418	0.338	0.544	0.251	0.373	0.380
0.513	0.608	0.499	0.523	0.413	1.766	0.476	0.386	0.620	0.287	0.425	0.433
0.520	0.615	0.506	0.530	0.418	0.476	1.781	0.391	0.627	0.290	0.430	0.439
0.421	0.498	0.410	0.429	0.338	0.386	0.391	1.585	0.508	0.235	0.349	0.355
0.677	0.801	0.658	0.690	0.544	0.620	0.627	0.508	1.966	0.378	0.560	0.571
0.313	0.370	0.304	0.319	0.251	0.287	0.290	0.235	0.378	1.380	0.259	0.264
0.464	0.549	0.451	0.473	0.373	0.425	0.430	0.349	0.560	0.259	1.611	0.392
0.473	0.560	0.460	0.482	0.380	0.433	0.439	0.355	0.571	0.264	0.392	1.691

· 10⁻⁵



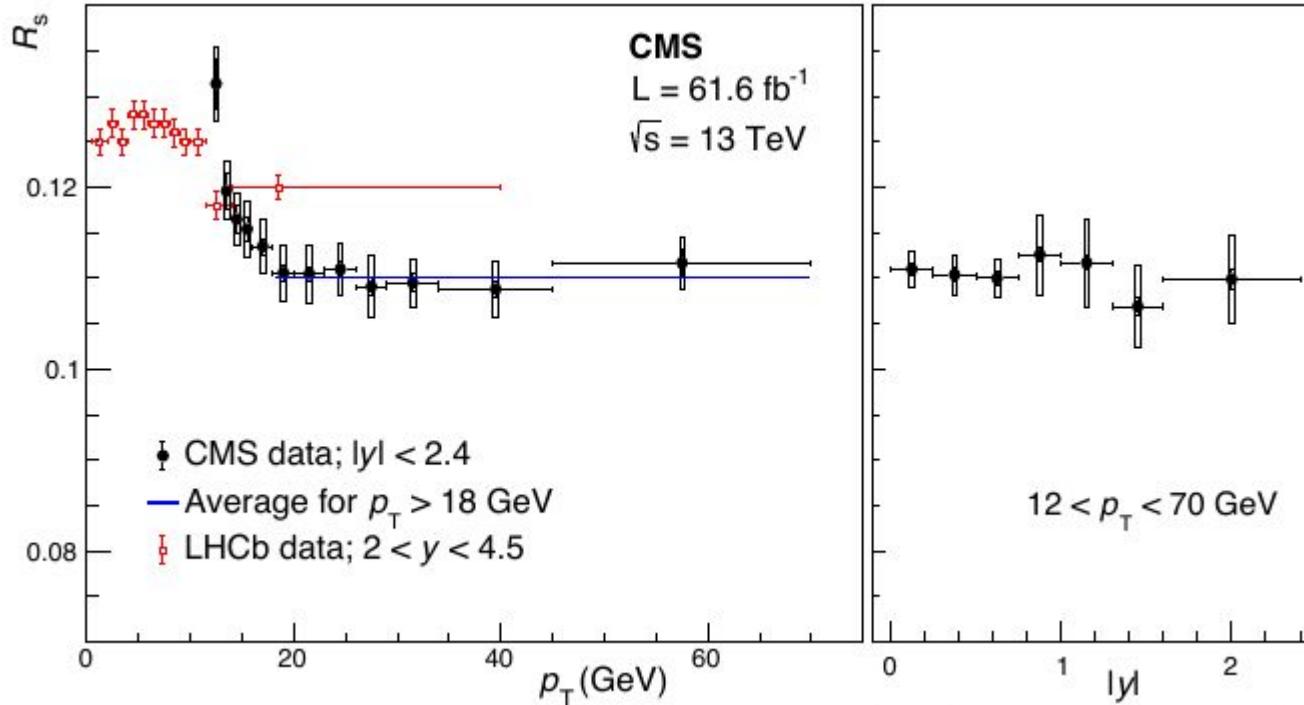
BPH-21-001

Motivation



[BPH-21-001](#)

Motivation



[BPH-21-001](#)

ROOT es muy flexible y proporciona una interfaz de programación para usar en aplicaciones propias y una interfaz gráfica de usuario para el análisis interactivo de datos.

Descripción general y justificación del curso:

Este curso pretende aportar a la formación de los estudiantes, aportando a sus conocimientos las técnicas y métodos computacionales necesarios para un mejor entendimiento de los procesos involucrados en el tratamiento de los datos procesados en el CERN en colisiones de partículas. Dada la magnitud de los datos tomados, se han desarrollado técnicas para el manejo de cantidades masivas de información. Se pretende que los estudiantes adquieran habilidades computacionales que le permitan desenvolverse en este medio, ya que es un requisito esencial para poder participar de proyectos relacionados con el área de física de partículas experimental. Adicional a esa idea, sin embargo, es importante decir que también de esta forma se garantiza que el estudiante tenga una formación integral tanto en las bases conceptuales aprendidas en la carrera como en las habilidades computacionales que le serán de gran utilidad enfrentando problemas tanto de tipo académico así como en el ámbito de la industria en el manejo de grandes cantidades de datos.

Objetivo general:

Adquirir conocimientos básicos y experiencia en el uso del marco de análisis de datos usando el software ROOT del CERN y adquirir habilidades para manejo de grandes volúmenes de datos con el mismo.

ROOT in a Nutshell

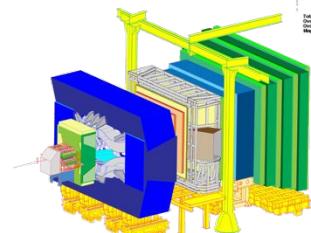
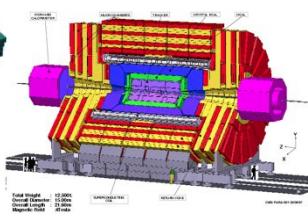
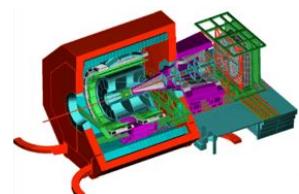
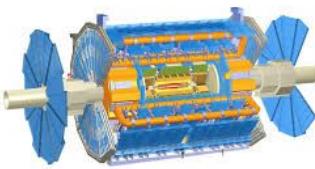
- ▶ ROOT is a software framework with building blocks for:
 - Data processing
 - Data analysis
 - Data visualisation
 - Data storage
- ▶ ROOT is written mainly in C++ (C++11/17 standard)
 - Bindings for Python available as well
- ▶ Adopted in High Energy Physics and other sciences (but also industry)
 - 1 EB of data in ROOT format
 - Fits and parameters' estimations for discoveries (e.g. the Higgs)
 - Thousands of ROOT plots in scientific publications

ROOT in a Nutshell

- ▶ ROOT can be seen as a collection of building blocks for various activities, like:
 - **Data analysis: histograms, graphs, functions**
 - **I/O: row-wise, column-wise** storage of any C++ object
 - **Statistical tools** (RooFit/RooStats): rich modeling and statistical inference
 - Math: **non trivial functions** (e.g. Erf, Bessel), optimised math functions
 - **C++ interpretation**: full language compliance
 - **Multivariate Analysis** (TMVA): e.g. Boosted decision trees, NN
 - **Advanced graphics** (2D, 3D, event display)
 - **Declarative Analysis**: RDataFrame

ROOT Application Domains

A selection of the experiments adopting ROOT



Event Filtering

Data

Offline Processing

Reconstruction

Further processing,
skimming

Analysis

Event Selection,
statistical treatment ...

Raw

Reco

Analysis
Formats

Images

Data Storage: Local, Network

Interpreter

- ROOT has a built-in interpreter : CLING
 - C++ interpretation: highly non trivial and not foreseen by the language!
 - One of its kind: Just In Time (JIT) compilation
 - A C++ interactive shell
- Can interpret “macros” (non compiled programs)
 - Rapid prototyping possible
- ROOT provides also Python bindings
 - Can use Python interpreter directly after a simple *import ROOT*
 - Possible to “mix” the two languages (????)

```
$ root
root[0] 3 * 3
(const int) 9
```

Persistency or Input/Output (I/O)

- ROOT offers the possibility to write C++ objects into files
 - This is impossible with C++ alone
 - Used the LHC detectors to write several petabytes per year
- Achieved with serialization of the objects using the reflection capabilities, ultimately provided by the interpreter
 - Raw and column-wise streaming
- As simple as this for ROOT objects: one method - *TObject::Write*

Cornerstone for storage
of experimental data

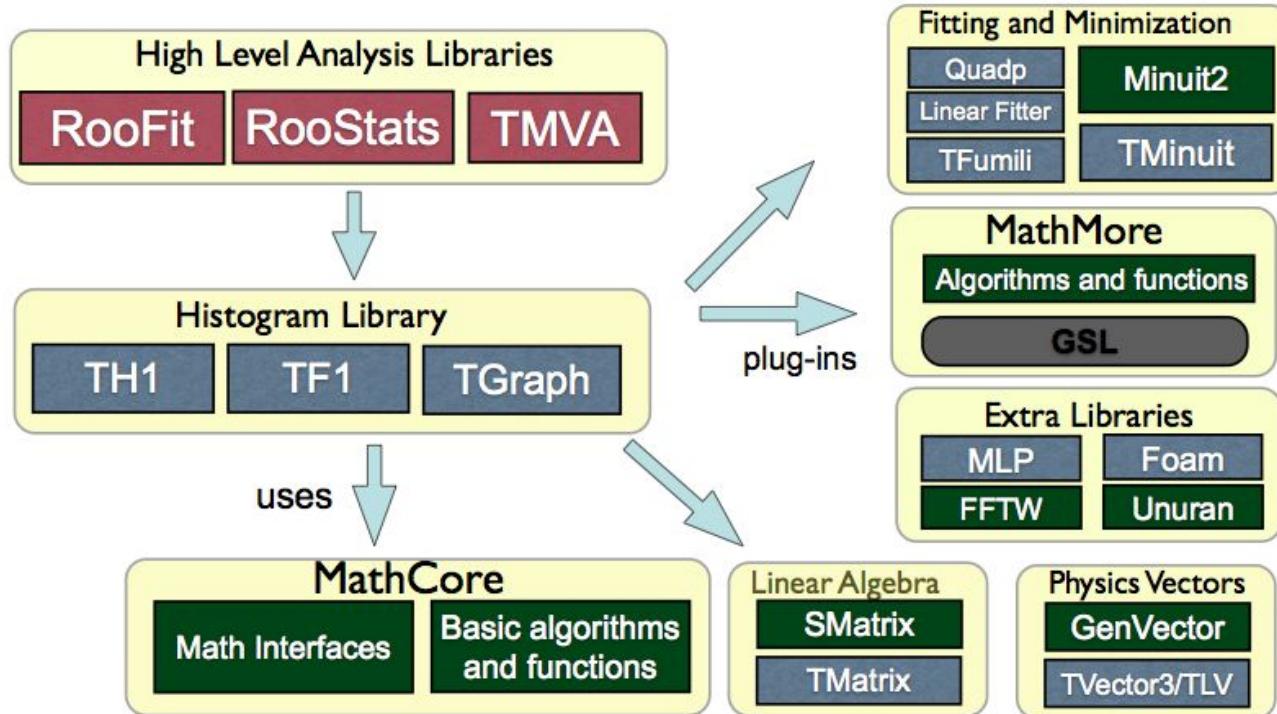
LHC Data in ROOT Format

1 EB

as of 2017

Mathematics and Statistics

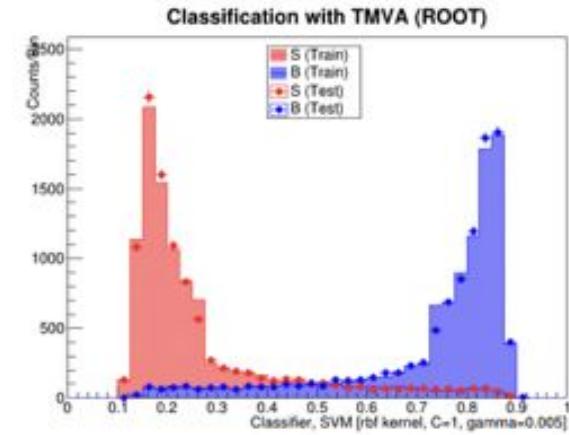
- ▶ ROOT provides a rich set of mathematical libraries and tools for sophisticated statistical data analysis



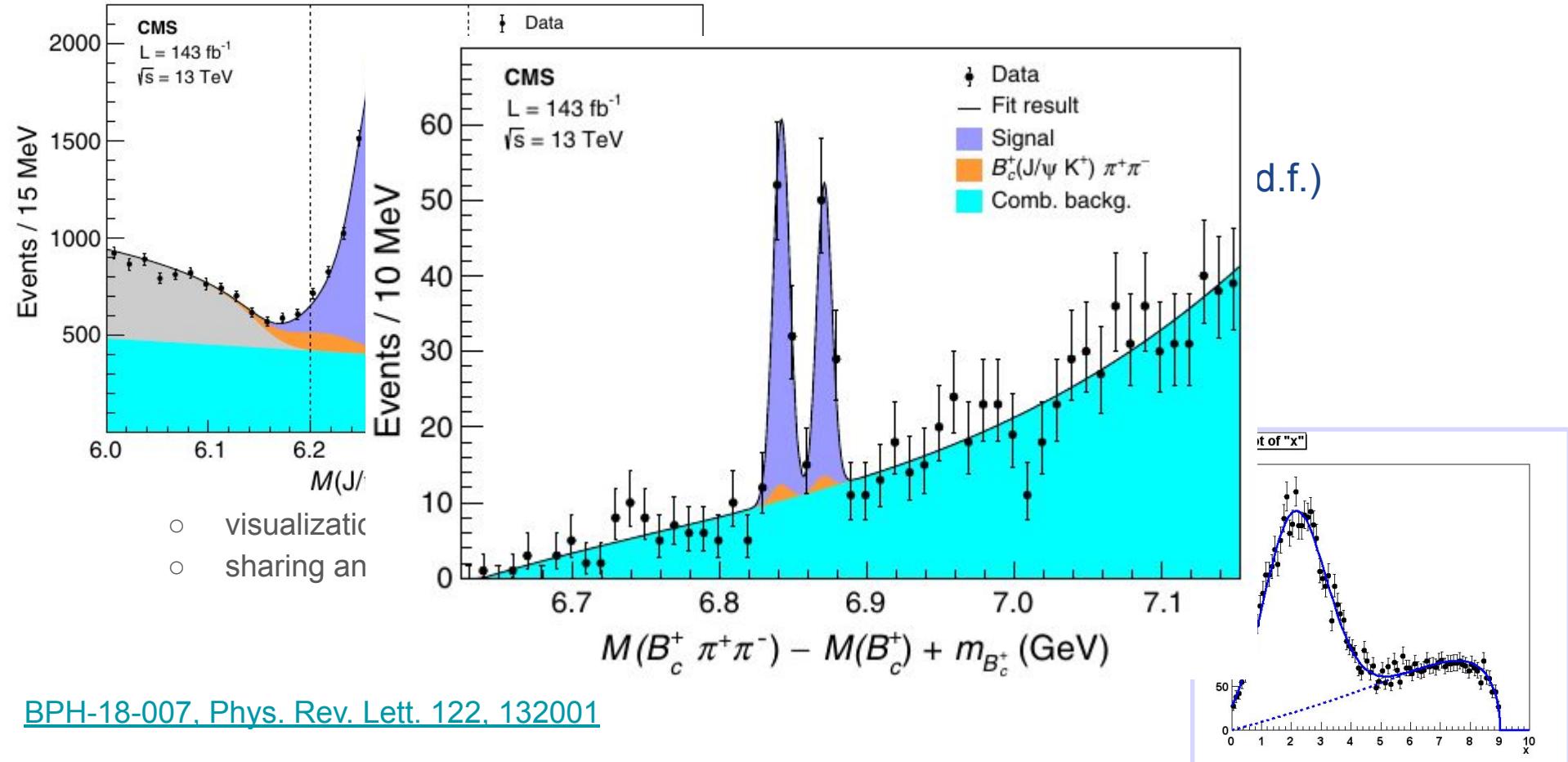
Machine Learning: TMVA

TMVA : Toolkit for Multi-Variate data Analysis in ROOT

- provides several built-in ML methods including:
 - Boosted Decision Trees
 - Deep Neural Networks
 - Support Vector Machines
- and interfaces to external ML tools
 - scikit-learn, Keras (Theano/Tensorflow), R

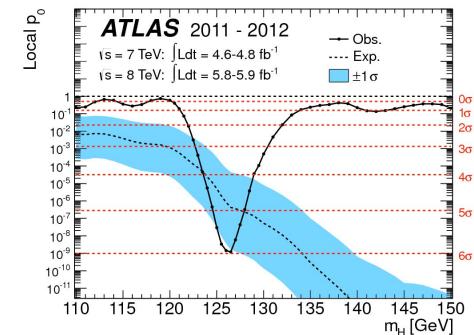
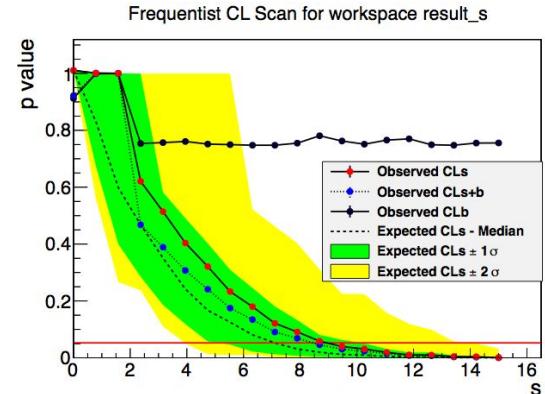


RooFit



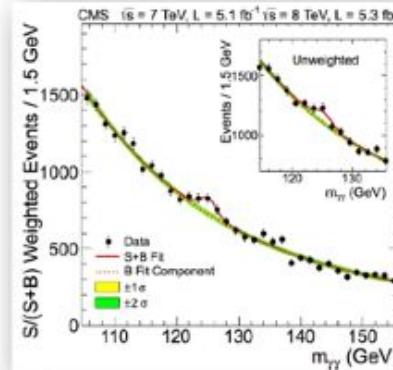
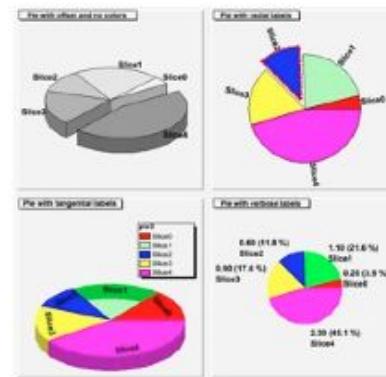
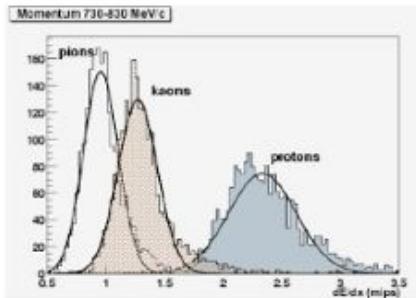
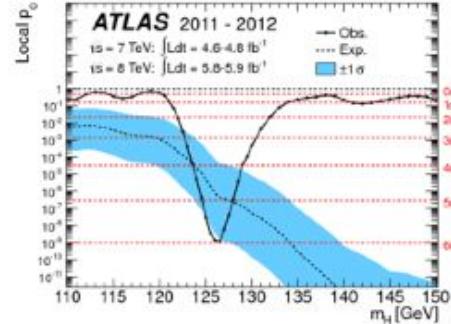
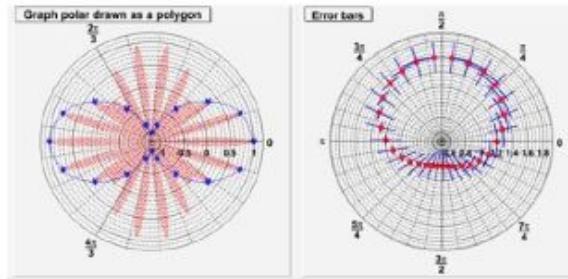
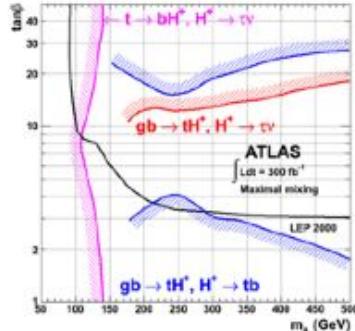
RooStats

- Advanced Statistical Tools for HEP analysis. Used for :
 - estimation of Confidence/Credible intervals
 - hypotheses Tests
 - e.g. Estimation of Discovery significance
- Provides both Frequentist and Bayesian tools
- Facilitate combination of results

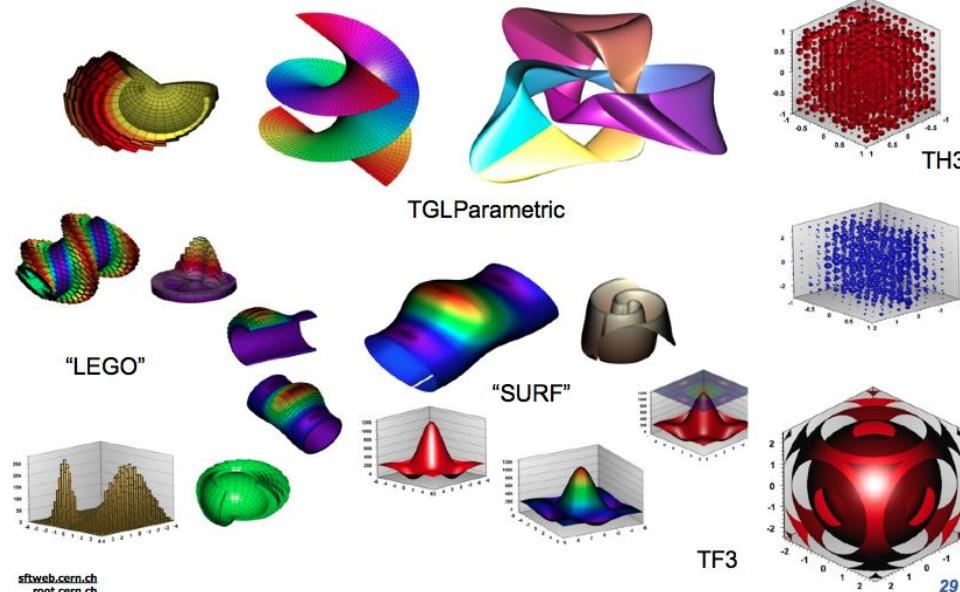


Graphics in ROOT

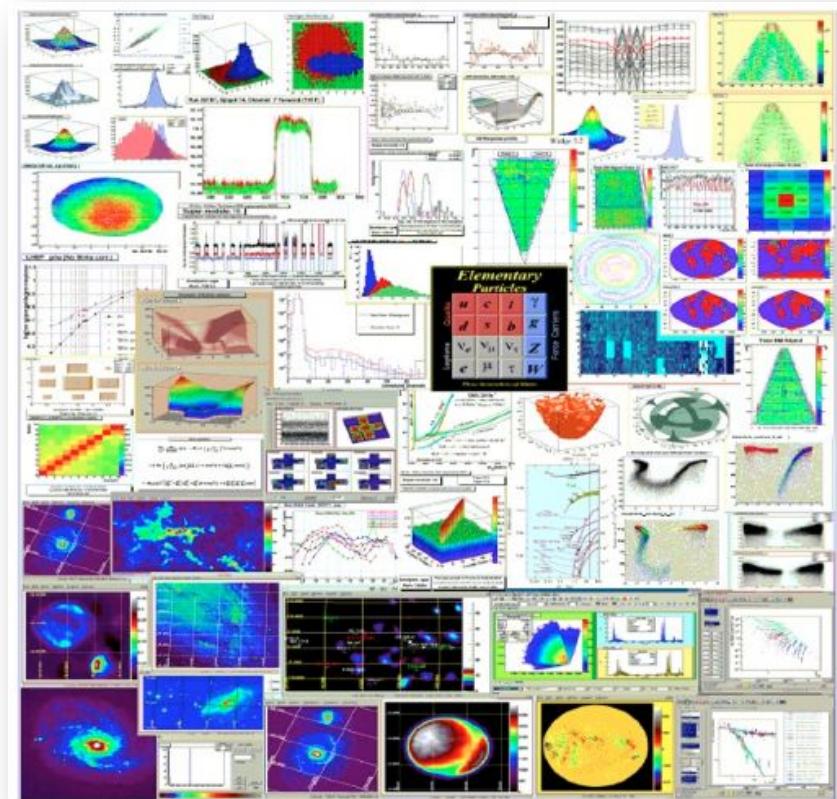
- ▶ Many formats for data analysis, and not only, plots



2D and 3D Graphics



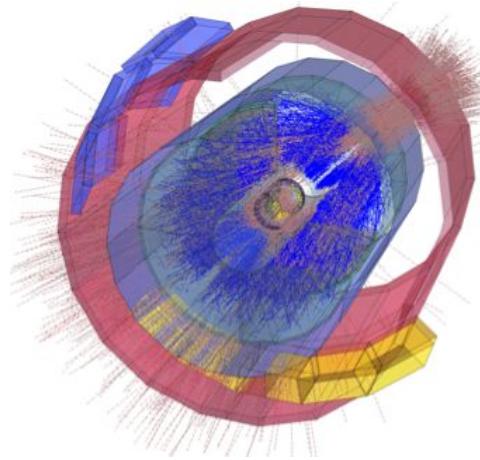
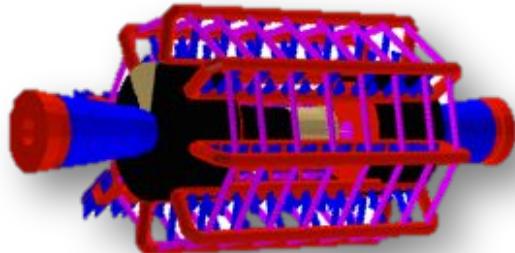
Can save graphics in many formats:
ps, pdf, svg, jpeg, LaTeX, png, c, root ...



Parallelism

- Many ongoing efforts to provide means for parallelisation in ROOT
- Explicit parallelism
 - **TThreadExecutor** and **TProcessExecutor**
 - Protection of resources
- Implicit parallelism
 - **TDataFrame**: functional chains
 - **TTreeProcessor**: process tree events in parallel
 - **TTree::GetEntry**: process of tree branches in parallel
- Parallelism is a fundamental element for tackling data analysis during LHC Run III and HL-LHC

Many More Features!



- ▶ Geometry Toolkit
 - Represent geometries as complex as LHC detectors
- ▶ Event Display (EVE)
 - Visualise particle collisions within detectors

The SWAN Service

- **Data analysis with ROOT “as a service”**
- *Interface:* Jupyter Notebooks
- *Goals:*
 - Use ROOT only with a web browser
 - Platform independent ROOT-based data analysis
 - Calculations, input and results “in the Cloud”
 - Allow easy sharing of scientific results: plots, data, code
 - Through your CERNBox
 - Simplify teaching of data processing and programming



<http://swan.web.cern.ch>

- ROOT web site: **the** source of information and help for ROOT users
 - For beginners and experts
 - Downloads, installation instructions
 - Documentation of all ROOT classes
 - Manuals, tutorials, presentations
 - Forum
 - ...

The screenshot shows the official ROOT website at <https://root.cern>. The header features the ROOT logo and navigation links for Download, Documentation, News, Support, About, Development, and Contribute. Below the header are four main links: Getting Started, Reference Guide, Forum, and Gallery, each with a corresponding icon. A section titled "ROOT is ..." provides a brief overview of the software's capabilities. A prominent "Download" button is highlighted with a red oval. The right side of the page includes a news feed with items like "Try it in your browser! (Beta)" and "Under the Spotlight" featuring ROOTbooks on Binder. There are also sections for "Other News" and "Latest Releases". At the bottom, there is a sitemap and footer links.

ROOT
Data Analysis Framework

Download Documentation News Support About Development Contribute

Getting Started Reference Guide Forum Gallery

ROOT is ...

A modular scientific software framework. It provides all the functionalities needed to deal with big data processing, statistical analysis, visualisation and storage. It is mainly written in C++ but integrated with other languages such as Python and R.

Try it in your browser! (Beta)

Download or Read More ...

Under the Spotlight

16-12-2015 Try the new ROOTbooks on Binder (beta)
05-01-2016 Wanted: A tool to warn user of inefficient (for I/O) construct in data model
03-12-2015 ROOT-TSeq::GetSize() or ROOT::seq::size()
02-09-2015 Wanted: Storage of HEP data via key/value storage solutions

Other News

16-04-2016 The status of reflection in C++
01-01-2016 Wanted: A tool to warn user of inefficient (for I/O) construct in data model
03-12-2015 ROOT-TSeq::GetSize() or ROOT::seq::size()
02-09-2015 Wanted: Storage of HEP data via key/value storage solutions

Latest Releases

Release 6.06/04 - 2016-05-03
Release 5.34/36 - 2016-04-05
Release 6.04/16 - 2016-03-17
Release 6.06/02 - 2016-03-03

SITEMAP

Download	Documentation	News	Support	About	Development	Contribute
Download ROOT All Releases	Reference Manual User's Guides Howto's Courses Building ROOT Patch Release Notes Code Examples	Blog Publications Workshops	Forum Bug submission Meetings Submit a Bug RootTalk Digest	Join us Contact Us Project Founders Team Previous Developers	Program of Work Release Checklist Project Statistics Coding Conventions Git Primer Browse Sources Meetings	Contributors Collaborate With Us

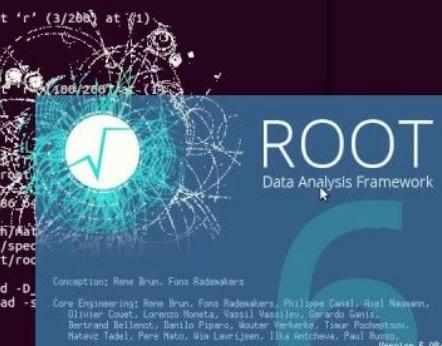
ROOT Basics

```
Terminal
g++ -m64 -O2 -DNDEBUG -Wl,--no-undefined -Wl,--as-needed -o bin/ssh2rpd main/src/ssh2rpd.o core/clib/src/snprintf.o
      -Lm -ldl -pthread -rdynamic
g++ -O2 -DNDEBUG -pipe -m64 -std=c++11 -Wshadow -Wall -W -Woverloaded-virtual -fPIC -Iinclude -pthread -MMD -MP -Iinspire-03@inspire03-OptiPlex-3040:~ root
g++ -m64 -O2 -DNDEBUG -Wl,--no-undefined -Wl,--as-needed -o bin/roots.exe main/src/roots.o \
      -Llib -Lcore -Lm -ldl -pthread -rdynamic
Install roots wrapper.
g++ -O2 -DNDEBUG -pipe -m64 -std=c++11 -Wshadow -Wall -W -Woverloaded-virtual -fPIC -Iinclude -pthread -MMD -MP -Iinspire-03@inspire03-OptiPlex-3040:~ root
g++ -m64 -O2 -DNDEBUG -Wl,--no-undefined -Wl,--as-needed -o bin/rootnb.exe main/src/nbmain.o \
      -Llib -Lcore -Lm -ldl -pthread -rdynamic
g++ -O2 -DNDEBUG -pipe -m64 -std=c++11 -Wshadow -Wall -W -Woverloaded-virtual -fPIC -Iinclude -pthread -MMD -MP -Iinspire-03@inspire03-OptiPlex-3040:~ root
g++ -m64 -O2 -DNDEBUG -Wl,--no-undefined -Wl,--as-needed -o bin/h2root main/src/h2root.o \
      -Llib -LRIO -LHist -LGrf -Lgraf3d -Lgrpad -Ltree -LMatrix -LNet -LThread -LMathCore -Lcore lib/libminicern.root [0]
      /usr/lib/gcc/x86_64-linux-gnu/5/libgcc.so /usr/lib/gcc/x86_64-linux-gnu/5/libcfortranbegin.a -Lm -ld
gfortran -O2 -DNDEBUG -fPIC -m64 -std=legacy -o main/src/g2root.o -c /home/inspire-03/SOFTWARE/root-6.08.02/main/src/g2root.f:412:30:
/home/inspire-03/SOFTWARE/root-6.08.02/main/src/g2root.f:412:30:
          call torels(3,pmixt,creals,ncr)
          ^
1
Warn-Ignore: Actual argument contains too few elements for dummy argument 'r' (3/20) at '1'
/home/inspire-03/SOFTWARE/root-6.08.02/main/src/g2root.f:728:29:
          call torels(-npars0,dummpars(1),creals,ncr)
          ^
1
Warn-Ignore: Actual argument contains too few elements for dummy argument 'r' (3/20) at '1'
/home/inspire-03/SOFTWARE/root-6.08.02/main/src/g2root.f:730:28:
          call torels(npars0,qjv(7),creals,ncr)
          ^
1
Warn-Ignore: Actual argument contains too few elements for dummy argument 'r' (3/20) at '1'
/home/inspire-03/SOFTWARE/root-6.08.02/main/src/g2root.f:731:28:
          call torels(npars0,qjv(7),creals,ncr)
          ^
1
Generating PCH for core/base/core/thread/io/io/math/mathcore/net/net/math/math/
math/physics/graf2d/postscript/core/rint/math/matrix/math/smatrix hist/spec/
2d/gviz bindings/pyroot math/genvector math/genvector math/minuit rootfit/roc
./bin/rootcling -rootbuild -l -f allDict.cxx -noDictSelection -c -pthread -D
clclude -I/usr/include/freetype2 -I/usr/include/graphviz -m64 -pipe -pthread -s
root []
Processing hsimple.c...
hsimple : Real Time = 0.12 seconds CPU Time = 0.07 seconds
(Tfile *) 0x240b300

=====
===== ROOT BUILD SUCCESSFUL.
===== Run 'source bin/thisroot.[c|sh]' before starting ROOT ===
=====

inspire-03@inspire03-OptiPlex-3040:~/SOFTWARE/root-6.08.02$ source ./bin/thisroot.sh
inspire-03@inspire03-OptiPlex-3040:~/SOFTWARE/root-6.08.02$ root
.....
| Welcome to ROOT 6.08/02 http://root.cern.ch |
| (c) 1995-2016, The ROOT Team |
| Built for linuxx86_64gcc |
| From tag v6-08-02, 2 December 2016 |
| Try '.help', '.demo', '.license', '.credits', '.quit'/.q' |

root [0].q
inspire-03@inspire03-OptiPlex-3040:~/SOFTWARE/root-6.08.02$ pwd
/home/inspire-03/SOFTWARE/root-6.08.02
inspire-03@inspire03-OptiPlex-3040:~/SOFTWARE/root-6.08.02$ 
```



Programación, diseño y complejidad.

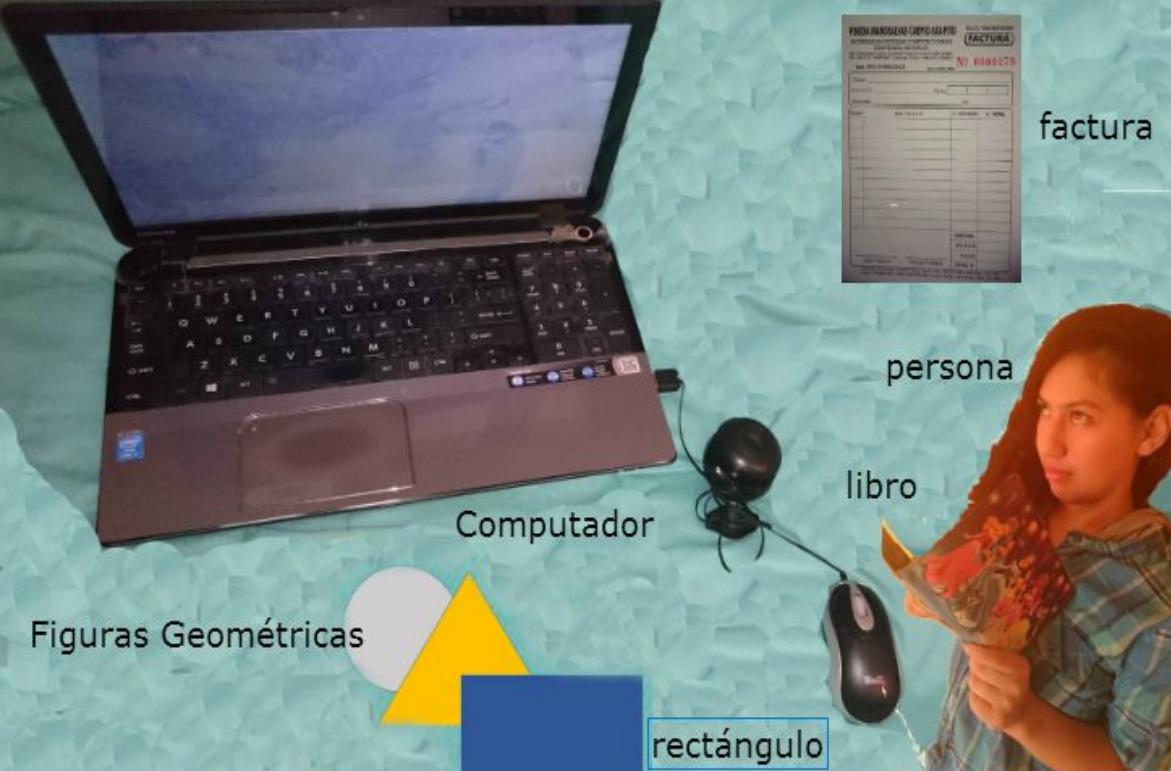
- **¿Cuál es el objetivo del software, resolver un problema particular?**
cálculo de problemas numéricos, mantenimiento de una base de datos organizada de información, búsqueda de nuevas partículas, análisis de imágenes...
- **En las últimas décadas, el crecimiento del poder computacional nos ha permitido abordar problemas cada vez más complejos**
- **Como consecuencia, el software también se ha vuelto más poderoso y complejo.**
Física no es la excepción: La colección de paquetes de software para la reconstrucción / análisis del experimento **BaBar** es ~ 6.4M líneas de C++.
“CMSSW , athena are written in C++ and Python, and has several million lines of code”.
- **¿Cómo lidiamos con una complejidad tan creciente?**

Filosofías de programación

- La clave para codificar con éxito sistemas complejos es descomponer el código en módulos más pequeños y minimizar las dependencias entre estos módulos.
- Los lenguajes de programación tradicionales (Fortran, Pascal, C) logran esto mediante los procedimientos orientados.
 - La modularidad y la estructura del software giran en torno a algoritmos encapsulados en "funciones"
 - Aunque las funciones son una herramienta importante en la estructuración de software, dejan algunos dolores de cabeza de diseño importantes
- Los lenguajes orientados a objetos (C++, Java, python ...) llevan estos pasos más allá

Agrupando datos y funciones asociadas en **objetos**.

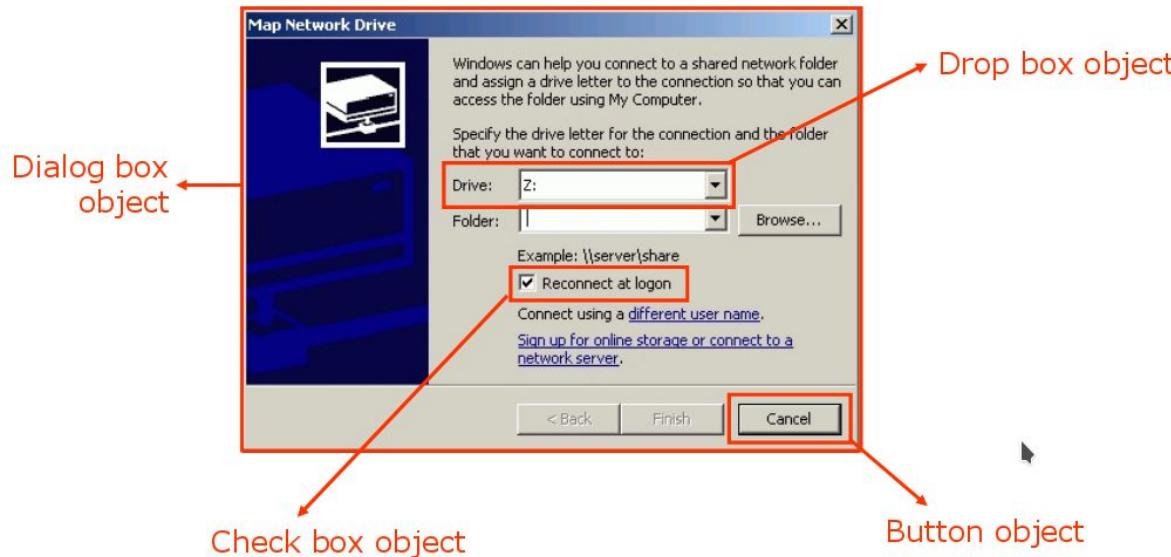




Identificar objetos concretos es relativamente sencillo, sin embargo identificar objetos abstractos requiere un poco de práctica e intuición como por ejemplo, transacción de una cuenta bancaria, detalle de una factura, que son objetos no tan evidentes, pero que son los que en la mayoría de casos serán los objetos que deberán ser implementados en programas, ya que interactuarán con los objetos más evidentes como la factura, o la cuenta bancaria en el caso de la transacción.

¿Qué son los objetos?

- ‘**Software objects**’ a menudo se encuentran de forma natural en problemas de la vida cotidiana.
- **Programación orientada a objetos (POO)** → Encontrar estos objetos y cual es su papel en su problema.

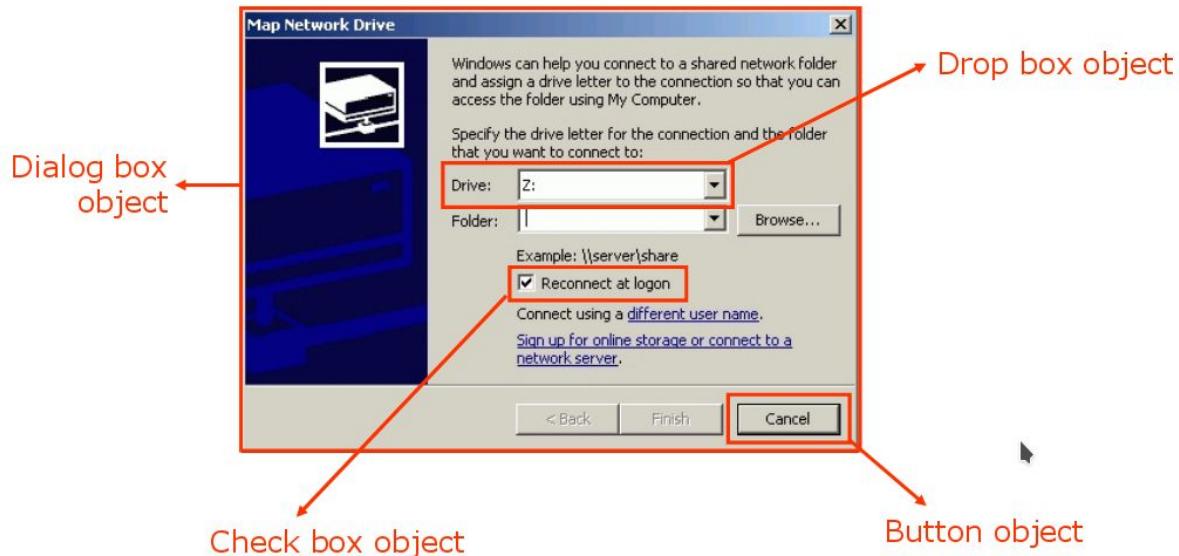


¿Qué son los objetos?

- Un objeto tiene:

Propiedades: posición, forma, etiqueta de texto

Comportamiento: si hace clic en el 'Cancel button', se produce una acción definida



El análisis y diseño orientado a objetos busca la relación entre objetos

- "Es-Un" Relacion (un objeto "PushButton" es un objeto Clickable).
- 'Tiene-Un' Relacion (a DialogBox Tiene Un CheckBox)

Beneficios de la programación orientada a objetos

- Beneficios de la programación orientada a objetos
 - **Reutilización del código existente** - los objetos pueden representar problemas genéricos.
 - Mantenimiento mejorado** - los objetos son más autónomos que las "subrutinas", por lo que el código está menos enredado.
 - A menudo, una forma "natural" de describir un sistema** - podemos ver el ejemplo anterior del "Dialog box"
- Pero....
 - El modelado orientado a objetos no sustituye al pensamiento sólido.
 - La POO **no garantiza un alto rendimiento**, pero **tampoco se interpone en su camino**.
- **Sin embargo**
 - POO es actualmente la mejor forma en que sabemos describir sistemas complejos.

Técnicas para lograr la abstracción

1. Modularidad 2. Encapsulación 3. Herencia 4. Polimorfismo.

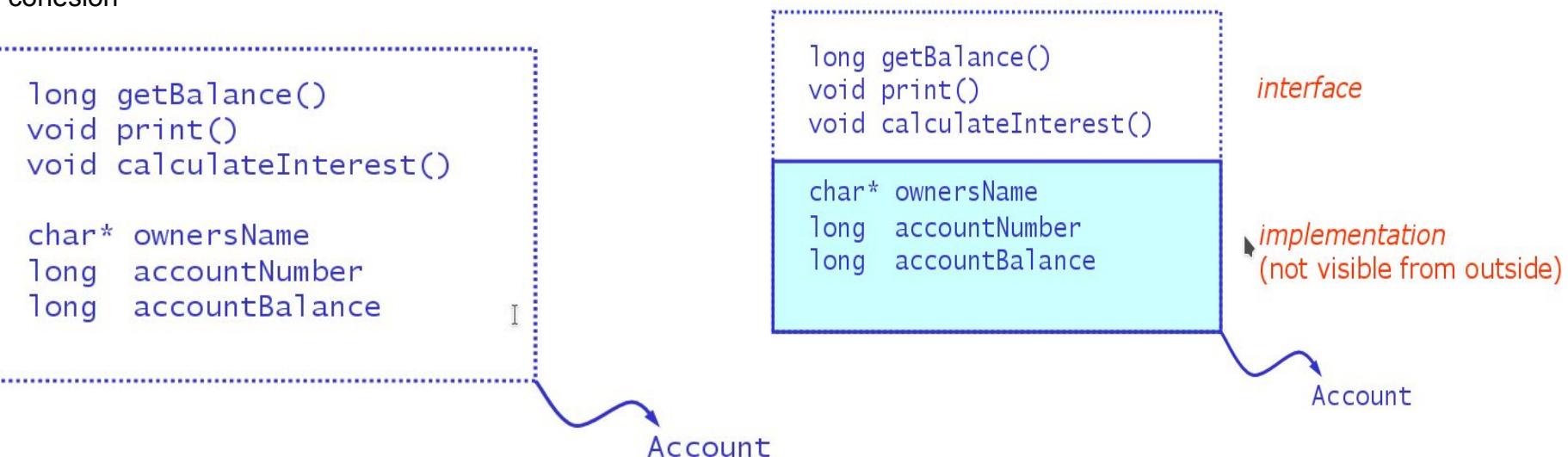
1. Descomponga su problema de forma lógica en unidades independientes

- Minimizar dependencias entre unidades - Acoplamiento suelto
- Agrupar cosas que tengan una conexión lógica - Fuerte cohesión

```
long getBalance()
void print()
void calculateInterest()
```

```
char* ownersName
long accountNumber
long accountBalance
```

2. Separar la interfaz y la implementación y proteja la implementación de los "usuarios" del objeto.

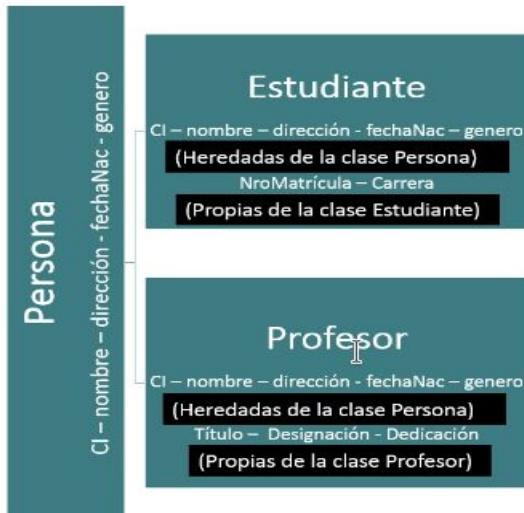


Técnicas para lograr la abstracción

1. Modularidad
2. Encapsulación
3. Herencia
4. Polimorfismo.

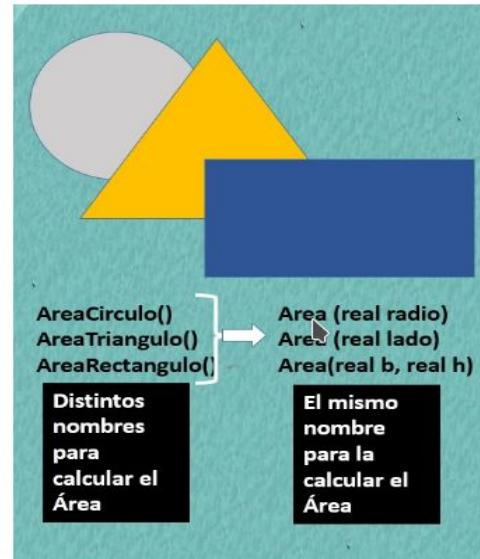
3. Herencia

Es el pilar más fuerte que asegura la reutilización de código, ya que a partir de esta característica es posible reutilizar (heredar) las características y comportamientos de una clase superior llamada clase padre, a sus clases hijas, denominadas clases derivadas.



4. Polimorfismo

A través de esta característica es posible definir varios métodos o comportamientos de un objeto bajo un mismo nombre, de forma tal que es posible modificar los parámetros del método, o reescribir su funcionamiento, o incrementar más funcionalidades a un método.



Introducción a C++

- Existe una amplia variedad de lenguajes POO: ¿por qué programar en C++?
 - Eso depende de para que lo necesitamos.
- **Ventaja de C++ - es un lenguaje compilado**
 - Cuando se usa correctamente, es el más rápido de todos los lenguajes POO
 - Debido a que las técnicas OO en C ++ se resuelven e implementan en tiempo de compilación en lugar de tiempo de ejecución. Entonces:
 - **Maximiza el rendimiento en tiempo de ejecución**
 - **No pagas por lo que no usas**
- **Desventaja de C++ - sintaxis más compleja**
 - Además, darse cuenta de la ventaja de rendimiento no siempre es trivial.
- C++ se utiliza mejor para proyectos a gran escala donde el rendimiento es importante
 - C++ se convirtió rápidamente en el estándar en High Energy Physics para el procesamiento de datos convencional, la adquisición de datos en línea, etc.
 - **Sin embargo, si el código de su programa será O (100) líneas y el rendimiento no es crítico, C, Python, Java pueden ser más eficientes.**

Disclaimer



Your tiny C++ intro:

- ▶ Variables, functions (incl overloading) and classes
- ▶ Pointers
- ▶ Scopes as lifetime of variables, and how to escape them (new/delete)
- ▶ Running code

This will be sufficient C++ to follow the ROOT-CERN course.

This is ***NOT*** sufficient C++ for real-life.

Bases de c++

Empezamos con un programa simple.

```
//Mi primer programa en C++
#include<iostream>

int main(){
    std::cout<< "Bienvenidos a c++! \n";
    return 0;
}
```

Bases de c++

Esta línea se corresponde con el comienzo de la definición de la función principal “**main**”. La función principal es el punto por donde todos los programas inician su ejecución, independientemente de su ubicación dentro del código fuente.

```
//Mi primer programa en C++
#include<iostream>
int main(){
    std::cout<< "Bienvenidos a c++! \n";
    return 0;
}
```

Utilice los objetos de la biblioteca iostream para imprimir cadenas a la salida estándar. Los nombres **std :: cout** y **std :: endl** se declaran en el “archivo de encabezado”.

Todas las líneas que comiencen con dos signos barra (**//**) se consideran comentarios y no tienen ningún efecto sobre el comportamiento del programa

Las líneas que comienza con un símbolo de “**#**” son directivas para el preprocesador. Este archivo específico (**iostream**) incluye las declaraciones de la norma básica de entrada y salida de la biblioteca de C++.

Esta declaración hace que la función principal termine. Un código de retorno es 0, cuando la función principal interpreta de manera general que el programa trabajó como se esperaba, sin ningún error durante su ejecución.

Variables y Tipos de datos

Variables Locales: Se definen solo en bloque en el que se vayan a ocupar, de esta manera evitamos tener variables definidas que luego no se utilizan.

Variables Globales: No son lo más recomendable, pues su existencia atenta contra la comprensión del código y su encapsulamiento.

Variables estáticas: Se tienen que inicializar en el momento en que se declaran, de manera obligatoria.

Los tipos de datos pueden ser predefinidos o abstractos. Un tipo de dato **predefinido** es intrínsecamente comprendido por el compilador. En contraste, un tipo de datos definido por el usuario es aquel que usted o cualquier otro programador crea como una clase, que comúnmente son llamados tipos de datos **abstractos**.

Los tipos de datos más comunes en C++ son:

TipodeDatos	EspacioenMemoria	Rango
unsigned char	8 bits	0 a 255
char	8 bits	-128 a 127
short int	16 bits	-32,768 a 32,767
unsigned int	32 bits	0 a 4,294,967,295
int	32 bits	-2,147,483,648 a 2,147,483,647
unsigned long	32 bits	0 a 4,294,967,295
enum	16 bits	-2,147,483,648 a 2,147,483,647
long	32 bits	-2,147,483,648 a 2,147,483,647
float	32 bits	3.4 x 10-38 a 3.4 x 10+38(6 dec)
double	64 bits	1.7 x 10-308 a 1.7*10+308(15 dec)
long double	80 bits	3.4 x 10-4932 a 1.1 x 10+4932
void	sin valor	

Decimal	Exponencial	cientifica
1625.0	1.625e3	1.625x10 ³
0.00731	7.31e-3	7.31x10 ⁻³

Definición de objetos de datos - variables

La definición de una variable se puede hacer de varias maneras.

```
int main() {  
    int j ; // definición - valor inicial indefinido  
    int k = 0 ; // definición con valor inicial  
    int l(0) ; // definición con inicialización de constructor  
    int L{0} ; // List initialization (introducido en C++11)  
  
    int m = k + l ; // el inicializador puede ser cualquier expresión C++ válida  
  
    int a,b=0,c(b+5); // declaración múltiple  
  
    return 0;  
}
```

```
int main() {  
    const float pi = 3.14159268 ; // objeto de datos constantes  
    pi = 2 ; // ERROR – no se compila  
}
```

Constantes en C++, const y #define

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

#define PI 3.1416; //Definimos una constante llamada PI

int main()
{
    std::cout << "Mostrando el valor de PI: " << PI << std::endl;

    return 0;
}
```

Error

Si intentamos ejecutar el código anterior obtendremos un error al haber usado el operador << justo después de PI, esto sucede porque PI no es tratado exactamente como una variable cualquiera sino como una expresión, así que realmente aunque podemos usar #define para declarar constantes no es la mejor opción.

bastante fácil y mejor aún ha sido mucho más intuitivo y sencillo. Se puede ver que la declaración es muy similar a la de una variable cualquiera y que ya no tenemos complicaciones al intentar añadir la instrucción endl para agregar el salto de línea

```
#include <iostream>

int main()
{
    const float PI = 3.1416;
    std::cout << "Mostrando el valor de PI: " << PI << std::endl;

    return 0;
}
```

Aritmetica

Operador modulo: $7\%4=3$, $17\%5 = 2$

(muchas aplicaciones, entre otras saber si un numero es par o no)

Reglas de precedencia

1. primero los parentesis
2. *, /, % se aplican despues (izquierda derecha)
3. Suma y Resta a lo ultimo (izquierda derecha)

$$\text{Álgebra: } m = \frac{a + b + c + d + e}{5}$$

$$C++: \quad m = (a + b + c + d + e) / 5;$$

$$\text{Álgebra: } z = pr \% q + w/x - y$$

$$C++: \quad z = p * r \% q + w / x - y;$$



$$y = a * x * x + b * x + c;$$



Operación en C++	Operador aritmético de C++	Expresión algebraica	Expresión en C++
Suma	+	$f + 7$	$f + 7$
Resta	-	$p - c$	$p - c$
Multiplicación	*	$b m \circ b \cdot m$	$b * m$
División	/	$x/y \circ \frac{x}{y} \circ x \div y$	x / y
Residuo	%	$r \bmod s$	$r \% s$

“using namespace std;”

El lenguaje C++ consta de un reducido número de instrucciones, pero ofrece un amplio repertorio de bibliotecas con herramientas que pueden ser importadas por los programas cuando son necesarias. Por este motivo, un programa suele comenzar por tantas líneas **#include** como bibliotecas se necesiten. Como se puede observar, en nuestro ejemplo se incluye la biblioteca **iostream**, necesaria cuando se van a efectuar operaciones de entrada (lectura de datos) o salida (escritura de datos). Para utilizar la biblioteca iostream es necesario utilizar el **espacio de nombres std**. Por ahora nos interesa recordar que nuestros programas pueden contener algunas de las siguientes directivas:

```
using namespace std::cout;  
using namespace std::cin;  
using namespace std::endl;
```

o, simplemente

using namespace std;

```
g++ -std=c++11 sumaenteros.C -o myco  
g++ -std=c++14 sumaenteros.C -o myco  
g++ sumaenteros.C
```

E2. Código: Suma de enteros

“Namespace”

- Permite agrupar funciones, clases, variables etc.. bajo un prefijo
- Los espacios de nombres se utilizan para evitar conflictos de nombres entre diferentes partes de un programa que podrían usar el mismo nombre para diferentes propósitos.
- Una definición de espacio de nombres comienza con la palabra clave ‘**namespace**’, seguida del nombre del espacio de nombres y un conjunto de llaves que encierran los miembros del espacio de nombres.
- Para llamar un elemento que pertenece a un espacio de nombre se usa el **operador ::**
- Para llamar un namespace de forma global se usa la palabra “using namespace”

E3. Código: **namespace.cxx**

Operadores

Operador	Tipo de Operador	Asociatividad			
[] -> .	Binarios	Izq. a Dch.	valor * valor	Producto	
! ~ - *	Unarios	Dch. a Izq.	valor / valor	División (
* / %	Binarios	Izq. a Dch.	valor % valor	Módulo (
+ -	Binarios	Izq. a Dch.	valor + valor	Suma	
<< >>	Binarios	Izq. a Dch.	valor - valor	Resta	
< <= > >=	Binarios	Izq. a Dch.			
== !=	Binarios	Izq. a Dch.	valor < valor	Comparación menor	
&	Binario	Izq. a Dch.	valor <= valor	Comparación menor o igual	
^	Binario	Izq. a Dch.	valor > valor	Comparación mayor	
	Binario	Izq. a Dch.	valor >= valor	Comparación mayor o igual	
&&	Binario	Izq. a Dch.	valor == valor	Comparación de igualdad	
	Binario	Izq. a Dch.	valor != valor	Comparación de desigualdad	
?:	Ternario	Dch. a Izq.			

Cin y asignacion

```
#include<iostream>

using namespace std;

int main(){
int number1; // primer entero
int number2; // segundo entero

cout << "entre dos numeros enteros: ";
cin >> number1 >> number2;

if ( number1 == number2 )
    cout << number1 << " == " << number2 << endl;

if ( number1 != number2 )
    cout << number1 << " != " << number2 << endl;

return 0;
}
```

```
#include<iostream>

using namespace std;

const int MAXIMO = 15 ;

int main(){

int cnt ;

cnt = 30 * MAXIMO + 1 ; // asigna a cnt el valor 451
cnt = cnt + 10 ; // asigna a cnt el valor 461

return 0;
}
```

E4. Código: compareenteros

Conteo

variable = variable + numero fijo;

i = i + 1 ; n = n + 2; m = m + 22;

El caso especial

i = i + 1 ; => i++ o ++i

```
int main(){  
in t ii = 0;  
  
cout << "contador = " << ii << endl;  
i++;  
cout << "contador = " << ii << endl;  
}
```

k=++n; => n=n+1; k=n;

k=n++; => k=n; n=n+1

Coerción

el valor de la expresión en el lado derecho del operador de asignación será convertido en el tipo de datos de la variable a la izquierda del operador de asignación.

int a = 25.9; // realmente en a se almacena 25

```
int b; float c;  
int d= b*c;
```

en el momento del calculo “b” y “c” son **double**, pero mantendrán su valor asignado (**int** y **float**) despues de eso. Ademas, aunque “b*c” es **double** “d” sera **int**.

Acumulacion

Sentencia	Equivalencia
<code>++variable;</code>	<code>variable = variable + 1;</code>
<code>--variable;</code>	<code>variable = variable - 1;</code>
<code>variable++;</code>	<code>variable = variable + 1;</code>
<code>variable--;</code>	<code>variable = variable - 1;</code>
<code>variable += expresion;</code>	<code>variable = variable + (expresion);</code>
<code>variable -= expresion;</code>	<code>variable = variable - (expresion);</code>
<code>variable *= expresion;</code>	<code>variable = variable * (expresion);</code>
<code>variable /= expresion;</code>	<code>variable = variable / (expresion);</code>
<code>variable %= expresion;</code>	<code>variable = variable % (expresion);</code>
<code>variable &= expresion;</code>	<code>variable = variable & (expresion);</code>
<code>variable ^= expresion;</code>	<code>variable = variable ^ (expresion);</code>
<code>variable = expresion;</code>	<code>variable = variable (expresion);</code>
<code>variable <<= expresion;</code>	<code>variable = variable << (expresion);</code>
<code>variable >>= expresion;</code>	<code>variable = variable >> (expresion);</code>

```
int sum = 0;  
sum = sum +1;  
sum = sum +95;
```



FUNCIONES MATEMÁTICAS

Nombre de la función	Descripción	Valor devuelto
abs(a)	valor absoluto	mismo tipo de datos que el argumento
pow(a1, a2)	a1 elevado a la potencia a2	tipo de datos del argumento a1
sqrt(a)	raíz cuadrada de un número real	precisión doble
sin(a)	seno de a (a en radianes)	doble
cos(a)	coseno de a (d en radianes)	doble
tan(a)	tangente de a (d en radianes)	doble
log(a)	logaritmo natural de a	doble
log10(a)	logaritmo común (base 10) de a	doble
exp(a)	e elevado a la potencia a	doble

4*sqrt(3.8*10.9-8.2)-7.6

```
#include<iostream>
#include<iomanip>
#include<cmath>
```

sqrt(cos(abs(theta)))

E5. Código: matfuncion

```
#include <iostream>
#include <cmath>

using namespace std;
#define PI 3.1416
int main()
{
    const float PI = 3.1416;
    float ag, ar, s,c,t;
    cout<< "Ingrese el angulo en grados" << endl;
    cin>>ag;
    ar = ag * PI /180;
    s = sin(ar);
    c = cos(ar);
    t = tan(ar);
    cout<<"El seno es "<<s<<endl;
    cout<<"El coseno es "<<c<<endl;
    cout<<"La tangente es "<<t<<endl;

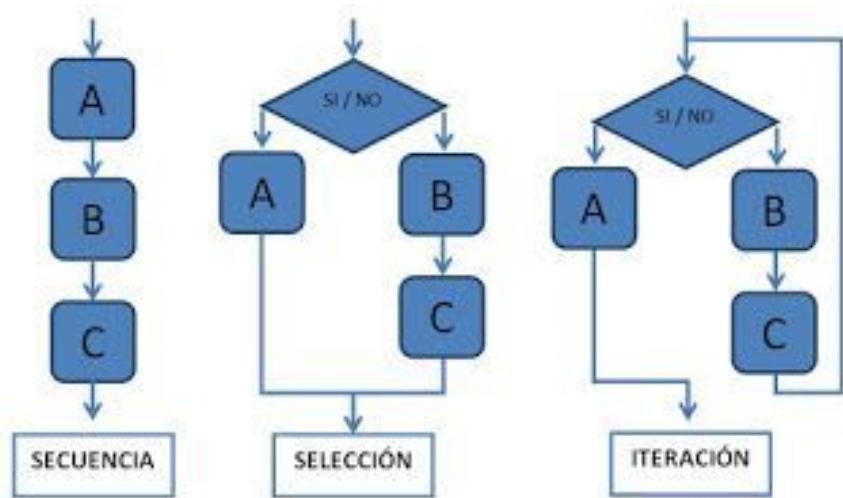
    cout<<" " << endl;
    cout<<"funciones log and exp "<< endl;
    cout<<" " << endl;

    cout<<"exp(1.0) "<<exp(1.0)<< endl;
    cout<<"log(10.0) "<<log(10.0)<< endl;
    cout<<"log(exp(1.0)) "<<log(exp(1.0))<< endl;
    cout<<"exp(2.30259) "<<exp(2.30259)<< endl;

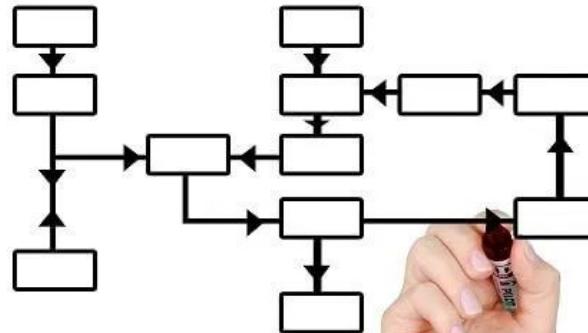
    //return 0;
}
```

Estructuras de control

- Estructuras de secuencia en cpp
- Instrucciones de selección en cpp
- Instrucciones de repetición en cpp



*Estructuras de control
en C++*



Ésta es la esencia de la simpleza

Cualquier programa de C++ que se desee crear puede construirse a partir de sólo siete tipos distintos de instrucciones de control (secuencia, if , if...else , switch , while , do...while y for), combinadas en sólo dos formas (apilamiento de instrucciones de control y anidamiento de instrucciones de control).

Criterios de selección. if-else

Si la calificación del estudiante es mayor o igual a 3.0
Imprimir “Aprobado”

De lo contrario

Imprimir “Rajado”

Si la condición produce cualquier valor numérico positivo o negativo diferente de cero, la condición es considerada como una condición “verdadera”

Condicion

```
if(calificacion>=3)  
cout << "Aprobado";  
else  
cout << "Rajado";
```

instrucción ejecutada si la condición es verdadera

Opcional

instrucción ejecutada si la condición es falsa;

```
cout << ( calificacionEstudiante >= 3 ? "Aprobado" : "Reprobado" );
```

E6. Código: compare2

Criterios de selección. if-else

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

int main(){
    int n1, porc;

    cout<< "Programa para determinar naturaleza par o impar de un numero" << endl;
    cout<<"Introduzca un numero entero: "<< endl;

    cin>> n1;

    porc= n1%2;

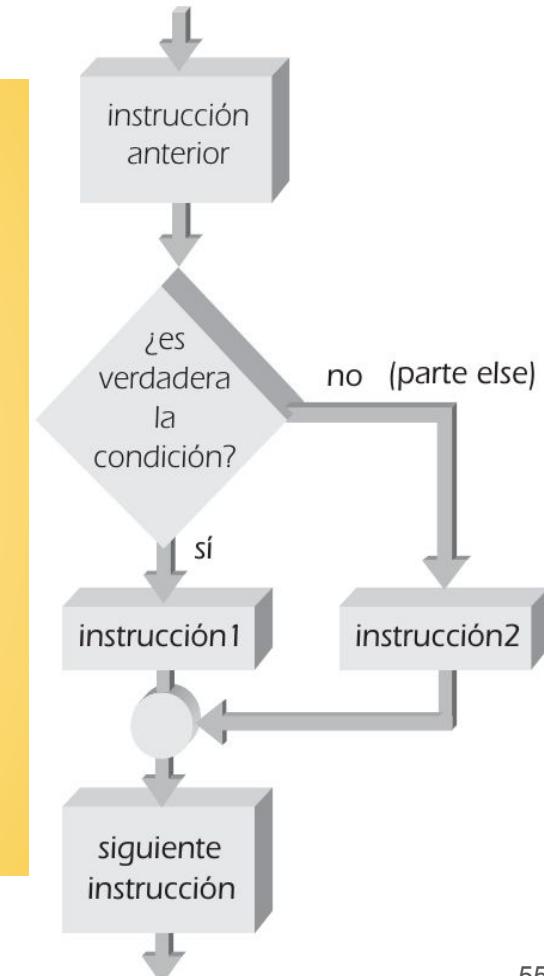
    if(porc==0)
        cout<< "El numero es par\n";
    else
        cout<< "El numero es impar\n";

    return 0;
}
```

E8. Código ifelse

E7. Código par.cpp (compuestos y anidados)
¿¿(if m==0){cout<<.....}?? tenga cuidado con el “==”

“Alcance de un
bloque”



Criterios de selección. La instrucción switch

```
switch(opción) //donde opción es la variable a comparar
{
    case valor1:
        Bloque de instrucciones 1;
        break;
    case valor2:
        Bloque de instrucciones 2;
        break;
    case valor3:
        Bloque de instrucciones 3;
        break;
    //Nótese que valor 1 2 y 3 son los valores que puede tomar la opción
    //la instrucción break es necesaria, para no ejecutar todos los casos.
    default:
        Bloque de instrucciones por defecto;
    //default, es el bloque que se ejecuta en caso de que no se de ningún caso
}
```

note:

case, break, default, switch

case: punto de partida

break: punto de terminación

default: caso opcional

Instrucción de repetición while

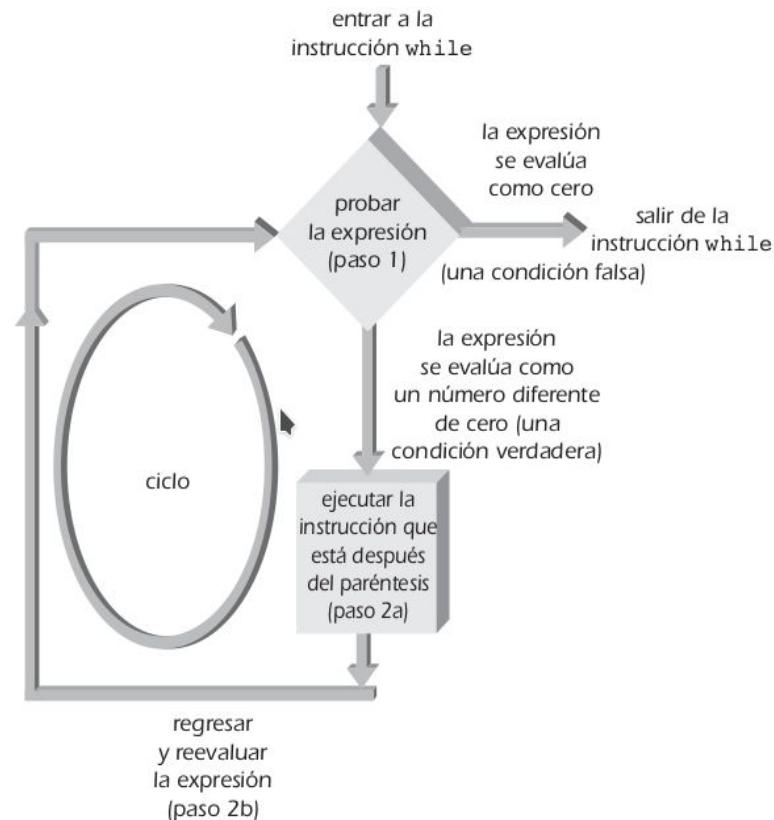
```
while(condición)
{
    grupo cierto de instrucciones;
    instrucción(es) para salir del ciclo;
}
```

```
#include <iostream>
using namespace std;
int main(){

    int producto = 3;
    while ( producto <= 100 ){
        producto = 3 * producto;
        cout << " el valor de producto es " << producto << endl;
    }

    return 0;
}
```

E10. Código testwhile



Tenga en cuenta que tenemos
Ciclos de cuenta variable y
Ciclos de cuenta fija.

Repetición controlada por un centinela

```
int main(){

    int n1, val;
    char ch;

    while (val != 1)
    {
        cout<< "entre un letra: ";
        cin >> ch;

        cout<< "entre un numero: ";
        cin >> n1;

        cout<< "¿desea entra mas letras y numeros? (NO = 1, SI = 2)";
        cin >> val;

        cout << " su letra y número son: "<< ch << " y " << n1 << endl;

    }

    return 0;
}
```

E11. Centinela.cpp

tenga cuidado con el tipo de datos

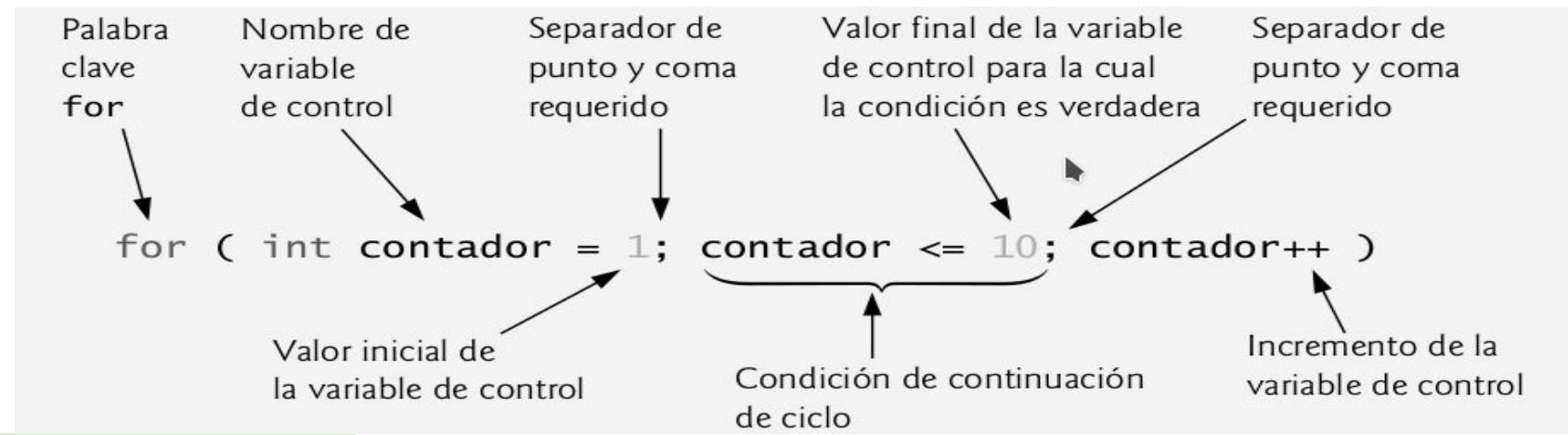
nota bene.

tenga cuidado con las expresiones
“break” y “continue”

break: rompe la ejecución del bucle de forma prematura sin terminar de ejecutar las demás órdenes

continue: permite saltar la ejecución de un bloque de código y continúa con la siguiente iteración.

Instrucción de repetición for



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

```
int main(){

    for ( int j = 1; j <= 10; ++ ){
        cout << j << endl;
    }

    return 0;
}
```

for (int j = x; j <= 4 * x * y; j += y / x)

for (int i = 100; i >= 1; i--) // 100 a 1 incrementos de -1

for (int i = 7; i <= 77; i += 7) // de 7 a 77 en incrementos de 7

for (int i = 99; i >= 0; i -= 11)// ¿ que hace esto?

Ciclos anidados

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

int main()
{
    int i=1,j;

    for(i; i <= 5;i++)
    {
        j=1;
        for (j; j <= i; j++ )
        {
            cout <<j;
        }
        cout << endl;
    }

    return 0;
}
```

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

int main()
{
    int i=1,j;
    while (i <= 5)
    {
        j=1;
        while (j <= i )
        {
            cout <<j;
            j++;
        }
        cout << endl;
        i++;
    }

    return 0;
}
```

E. example1_switch
ver todos los examples “loop_*.”

Ciclo do_while

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

int main()
{
int contador = 1;

do
{
    cout << contador << " ";
    contador++;
} while ( contador <= 10 );

cout << endl;

return 0;
}
```

do
instrucción
while (*condición*);

backup



Palabras clave de C++

Palabras clave comunes para los lenguajes de programación C y C++

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

Palabras clave sólo de C++

and	and_eq	asm	bitand	bitor
bool	catch	class	compl	const_cast
delete	dynamic_cast	explicit	export	false
friend	inline	mutable	namespace	new
not	not_eq	operator	or	or_eq
private	protected	public	reinterpret_cast	static_cast
template	this	throw	true	try
typeid	typename	using	virtual	wchar_t
xor	xor_eq			

Un paréntesis util (1)

Expresión	Valor	Interpretación
'A' > 'C'	0	falso
'D' <= 'Z'	1	verdadero
'E' == 'F'	0	falso
'g' >= 'm'	0	falso
'b' != 'c'	1	verdadero
'a' == 'A'	0	falso
'B' < 'a'	1	verdadero
'b' > 'z'	1	verdadero

“Hola” > “hola” (Falsa??)

“Bejuco” > “Beato” (verdadero??)

“Planta” > “Planeta” (verdadera??)

```
cout << "El valor de 3 < 4 es " << (3 < 4) << endl;
cout << "El valor de 2.0 > 3.0 es " << (2.0 > 3.3) << endl;
cout << "El valor de verdadero es " << verdadero << endl;
cout << "El valor de falso es " << falso << endl;
```

Un paréntesis util (2)

$a = 12.0$, $b = 2.0$, $i = 15$, $j = 30$ y $completo = 0.0$

Expresión	Valor	Interpretación
$a > b$	1	completo
$(i == j) \mid\mid (a < b) \mid\mid completo$	0	falso
$(a/b > 5) \&\& (i \leq 20)$	1	verdadero

Operadores relacionales:

$>$, $<$, $=$, \geq , \neq

Operadores lógicos:

$\&\&$, $\mid\mid$, $!$

mayor precedencia de los operadores relacionales en relación con los operadores lógicos

Un problema de exactitud numérica

$(a==b)?$

tenga cuidado con los “números” float y double



$abs(a-b) < 0.0000?$