

# Root 2008

[www.if.usp.br/suaide](http://www.if.usp.br/suaide)

Alexandre Suaide

aula 1

# Ferramentas para análise de dados

## • Paradigma 1

- Ferramentas prontas para executar tarefas específicas para análise de dados
  - Origin
  - Excel
  - MathCad
  - Mathematica
  - DAMM
  - SCAN
  - Etc
- A flexibilidade é limitada ao que cada programa oferece

## • Paradigma 2

- Ferramentas que permitem cada usuário criar o próprio ambiente de análise de dados
  - Linguagens de programação
  - PAW
  - ROOT
- A flexibilidade é muito elevada
- custo de uma intervenção maior do usuário

## Objetivos deste mini-curso

- **Mostrar os fundamentos do c++**
  - Não é um curso de lógica de programação
  - Programação orientada a objetos
- **Introdução ao ROOT**
  - ROOT como ferramenta de programação e análise de dados
- **4 aulas**
  - Ver <http://www.dfn.if.usp.br/~suaide>

# Programa

- **Aula 1**
  - Introdução ao c++ e ROOT
  - Conceito de classe e objeto
  - Básico de gráficos e funções no ROOT
- **Aula 2**
  - Mais gráficos e funções
  - Histogramas de 1 e 2D
  - Ajustes de funções, legendas, etc.
  - Escrevendo programas simples: Monte Carlo e simulações
- **Aula 3**
  - Referências e ponteiros
  - Nomes e memória
  - Programação mais complexa: mais Monte Carlo
- **Aula 4**
  - I/O no ROOT
  - Mais programação no ROOT
  - Compilando com o ROOT

# Instalando o ROOT

- Download na web
  - <http://root.cern.ch>
- Versões pré-compiladas
  - Windows
  - Mac OSX
  - Alguns Linux
- Fonte disponível
  - Possível compilar em outros sistemas



The screenshot shows the ROOT System Home Page in a web browser. The browser's address bar displays <http://root.cern.ch/root/>. The page features a navigation menu on the left with categories like Informatica, Physics, ROOT, Blogs, and MAC. The main content area includes a sidebar with links such as Roadmap, Mission Statement, Architecture, Main Features, CINT, Coding Conventions, Benchmarking, Picture Gallery, Publication List, and The ROOT Team. The central part of the page displays the ROOT logo in large blue letters, followed by the text "An Object-Oriented Data Analysis Framework" and an illustration of a woman holding a glowing ring. A prominent green banner announces a "Development release 5.19/04" dated "14/05/2008". Below this, a text box provides details about the release, including instructions for upgrading and links to release notes and download options. The footer of the page contains the URL <http://root.cern.ch/wiki/bin/view/ROOT/Download>.

# O que é ROOT? (<http://root.cern.ch>)

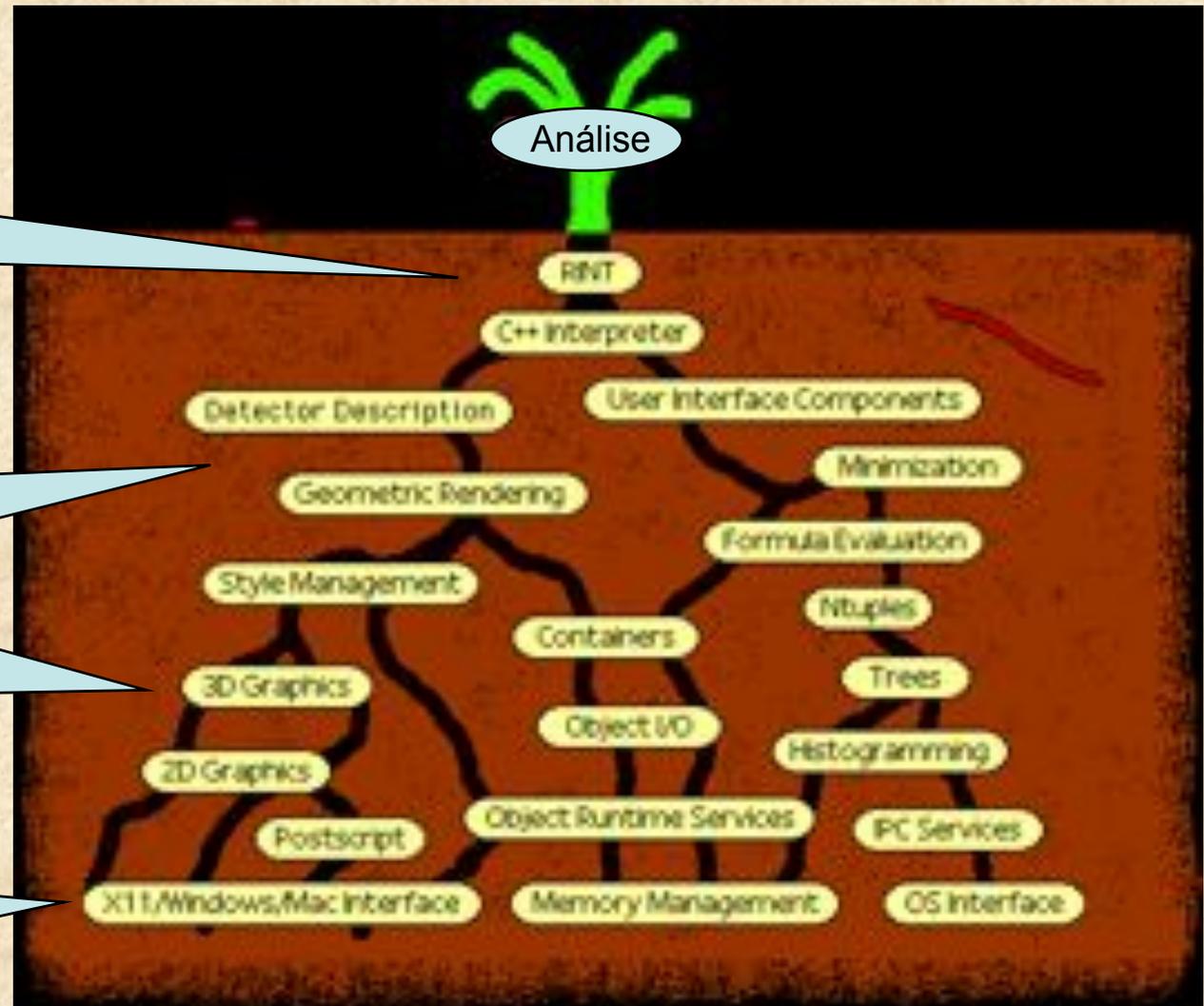
Uma análise sólida se constrói a partir de métodos confiáveis que não imponham limitações à criatividade do pesquisador

Interface entre o usuário e as ferramentas de análise

Ferramentas de visualização e manipulação gráfica

Ferramentas básicas I/O, ferramentas gráficas, métodos de cálculo

Nível mais fundamental. Interface com S.O.



## Então, o que é ROOT?

- Conjunto de bibliotecas escritas em c++ cuja finalidade é permitir o desenvolvimento de técnicas de simulação, aquisição e análise de dados
  - As bibliotecas seguem a filosofia de programação orientada a objeto
- A interface com o usuário se faz de três modos
  - Prompt de comando
    - Interpretador c/c++ (CINT)
      - Permite total acesso a funcionalidade do ROOT e c++
      - O prompt de comando atua como um compilador em tempo real.
  - Interface gráfica
    - Permite manipulação de objetos gráficos (histogramas, gráficos, objetos, menus, etc)
  - Compilador c++ (gcc em Linux e VC em windows)
    - Permite compilar programas avançados e criar novos programas específicos, utilizando a funcionalidade do ROOT
      - ScanRoot, SPMRoot
- Necessita conhecimento de c++ para fazer bom proveito do sistema

## Comandos básicos

- Como iniciar o programa 😊
  - Duplo clique no ícone
  - Linha de comando: digite `root`
- Como sair do ROOT
  - Digite `.q`
    - Estranho, mas como o ROOT é um interpretador c++, os comandos internos do mesmo têm que ser diferenciados. Assim, todos os comandos do ROOT começam com ".". Os comandos mais importantes, além do `.q` são
      - `.L` para carregar um programa (macro) na memória
      - `.x` para carregar e executar um programa
      - `.h` para um help dos comandos disponíveis

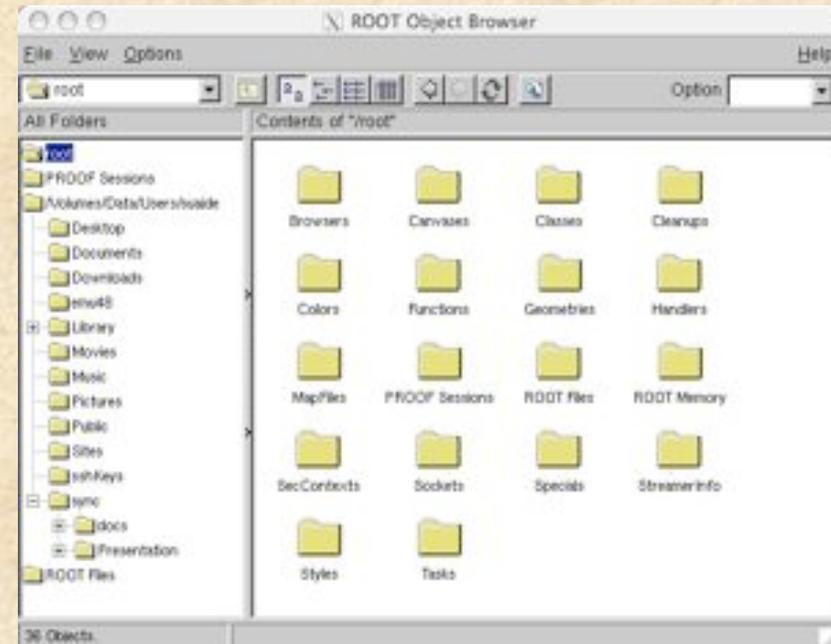
# A interface do ROOT

new TBrowser

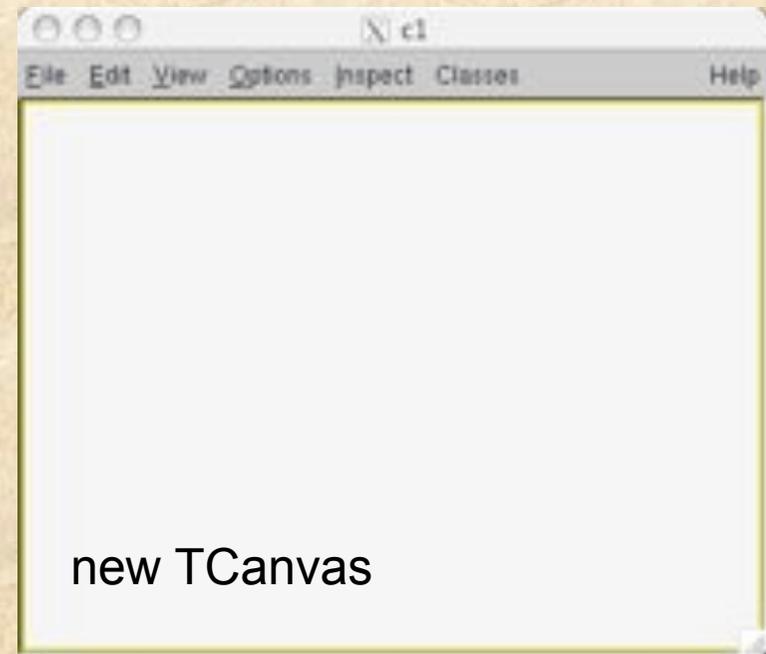
```
Terminal — root.exe — 80x24
[suasid@macbook:~]$ root
*****
*          WELCOME to ROOT          *
*          *          *          *
* Version  5.17/04  16 October 2007 *
*          *          *          *
* You are welcome to visit our Web site *
*          http://root.cern.ch        *
*****

ROOT 5.17/04 (trunk@20365, Oct 16 2007, 11:01:04 on macosx)

CENT/ROOT C/C++ Interpreter version 5.16.26, Oct 11, 2007
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] []
```



- Use a tecla TAB para obter ajuda
- `root [0] b = new TB <TAB>`
- `root [1] b = new TBrow<TAB>`
- `root [2] b = new TBrowser(<TAB>`
- Útil para descobrir a lista de métodos
- Descobrir também lista de parâmetros



# ROOT é um interpretador C++ online

- Qualquer comando c++ digitado no prompt é executado → programação online



```
Terminal — root.exe — 80x27

[tux@ideamacbook:~]$ root
*****
*                               *
*      W E L C O M E  t o  R O O T      *
*                               *
*  Version   5.19/04      14 May 2008   *
*                               *
*  You are welcome to visit our Web site *
*      http://root.cern.ch              *
*                               *
*****

ROOT 5.19/04 (trunk:823834, May 14 2008, 16:04:00 on macosx)

CINT/ROOT C/C++ Interpreter version 5.16.29, Jan 08, 2008
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }-
root [0]
root [0] float a = 9;
root [1] if(a>4) printf("-----> a > 4 \n");
-----> a > 4
root [2] for(int i = 0; i<3; i++) printf("i = %d\n",i);
i = 0
i = 1
i = 2
root [3] []
```

# Meu primeiro programa em c++

- Utilize o seu editor de textos favorito (vi, nedit, emacs, etc.)
- Entre no root e digite

```
root [0]: .x hello.C
Meu primeiro programa
root [1]: _
```

Tipo de retorno da função

Nome da função. Para ser executada com o comando .x deve ter o mesmo nome do arquivo

```
void hello()
{
    printf("Meu primeiro programa\n");
    return;
}
```

Arquivo hello.C

Separação de comandos são feitas através do ;

Funções ou blocos de programas são encapsulados com { }

# Variáveis

- Float, int, char, etc...
- No ROOT TAMBÉM: Float\_t, Int\_t, Char\_t
  - Para compensar diferenças entre sistemas

- Exemplos de definição de variáveis

`int a;` Definição simples

`int i,j,k;` Definição múltipla

`float pi=3.1415, r0=1.2;` Definição c/inicialização

`double a[10], b[5][20];` Definição de vetores

`float c[5] = {1,2,3,4,5};` Definição c/inicialização

`float d[3][2] = {{1,2,3},{4,5,6}};`

`char texto[20] = "testando";`

# Loops

- **for ( cond. inicial; condição de teste; alteração) { comandos }**

```
for(int i=0; i<10; i++)
{
    int a = i*i;
    cout << i << "*" << i << " = " << a << endl;
}
```

- **do { alguma coisa } while ( condição de teste );**

```
int i = 0;
do
{
    int a = i*i;
    cout << i << "*" << i << " = " << a << endl;
    i++;
} while (i<10);
```

- **while ( condição de teste ) { alguma coisa }**

```
int i = 0;
while (i<10)
{
    int a = i*i;
    cout << i << "*" << i << " = " << a << endl;
    i++;
}
```

# if ... else

- **Condições**

```
if ( condição ) comando;  
else outroComando;
```

O comando **else**  
é opcional

```
if ( condição )  
{  
    comando1;  
    comando2;  
}  
else  
{  
    comando1;  
    comando2;  
}
```

```
if (a<10) cout << "a é menor que 10" << endl;  
else cout << "não é menor que 10" << endl;  
  
if(a>10 && a<20) cout <<"a entre 10 e 20"<<endl;  
  
if(a!=10) cout <<"a é diferente de 10"<<endl;
```

- **Operadores condicionais**

== (igual), != (diferente), || (or), && (and), ! (not), etc.

## Funções em c++

- Funções permitem a compactação de um programa, fazendo com que blocos repetitivos sejam tratados de forma isolada

```
tipo nome(parametros)
{
    comandos;
    return valor;
}
```

- Exemplo: cálculo da área de um retângulo

```
float area(float lado1, float lado2)
{
    float a = lado1*lado2;
    return a;
}
float l1,l2;
cout << "Entre os lados do retângulo ";
cin >> l1 >> l2;
cout << "A área desse retângulo é " << area(l1,l2) <<endl;
```

# Polimorfismo

- Uma característica interessante em c é o polimorfismo. É possível definir funções com mesmo nome mas parâmetros distintos.
  - O c se encarrega de decidir qual função chamar.

- Exemplo: cálculo da área de um retângulo e quadrado

```
float area(float lado1, float lado2)
{
    float a = lado1*lado2;
    return a;
}
float area(float lado)
{
    float a = lado*lado;
    return a;
}
...
float area1 = area(10,20);
float area2 = area(10);
```

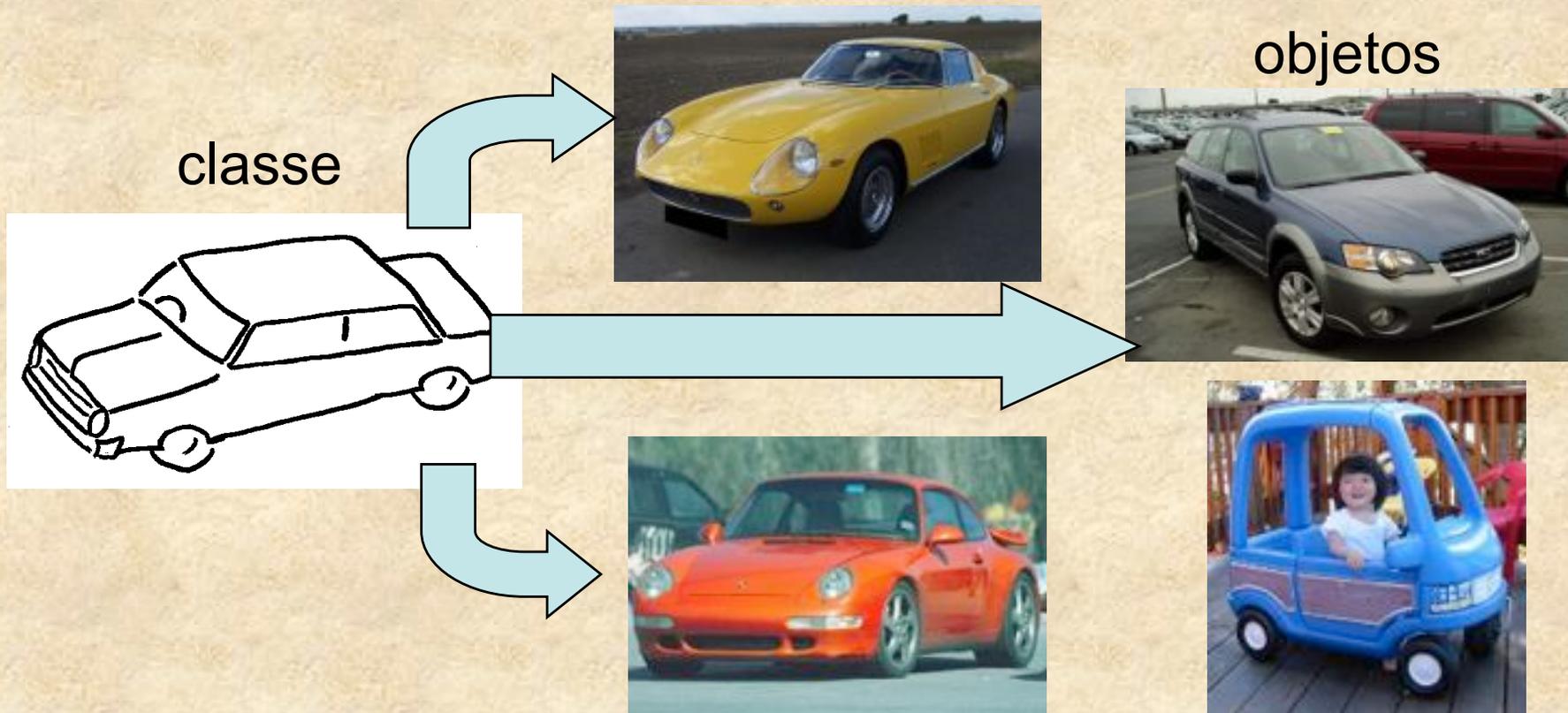
# Linguagens orientadas a objetos.

## Classes em c++

- **Classe**
  - É a descrição de algum conceito abstrato
  - Classes possuem membros
    - Membros podem ser variáveis ou funções.
    - Os membros podem ser públicos, privados ou protegidos. O usuário tem acesso somente aos membros públicos
  - Classes podem derivar de outras classes.
- **Objeto**
  - É a "coisa" real criada na memória do computador que tem sua estrutura definida pela classe
- **Ponteiro**
  - Contém o endereço de um objeto na memória
    - É uma espécie de atalho para uma certa região de memória do computador

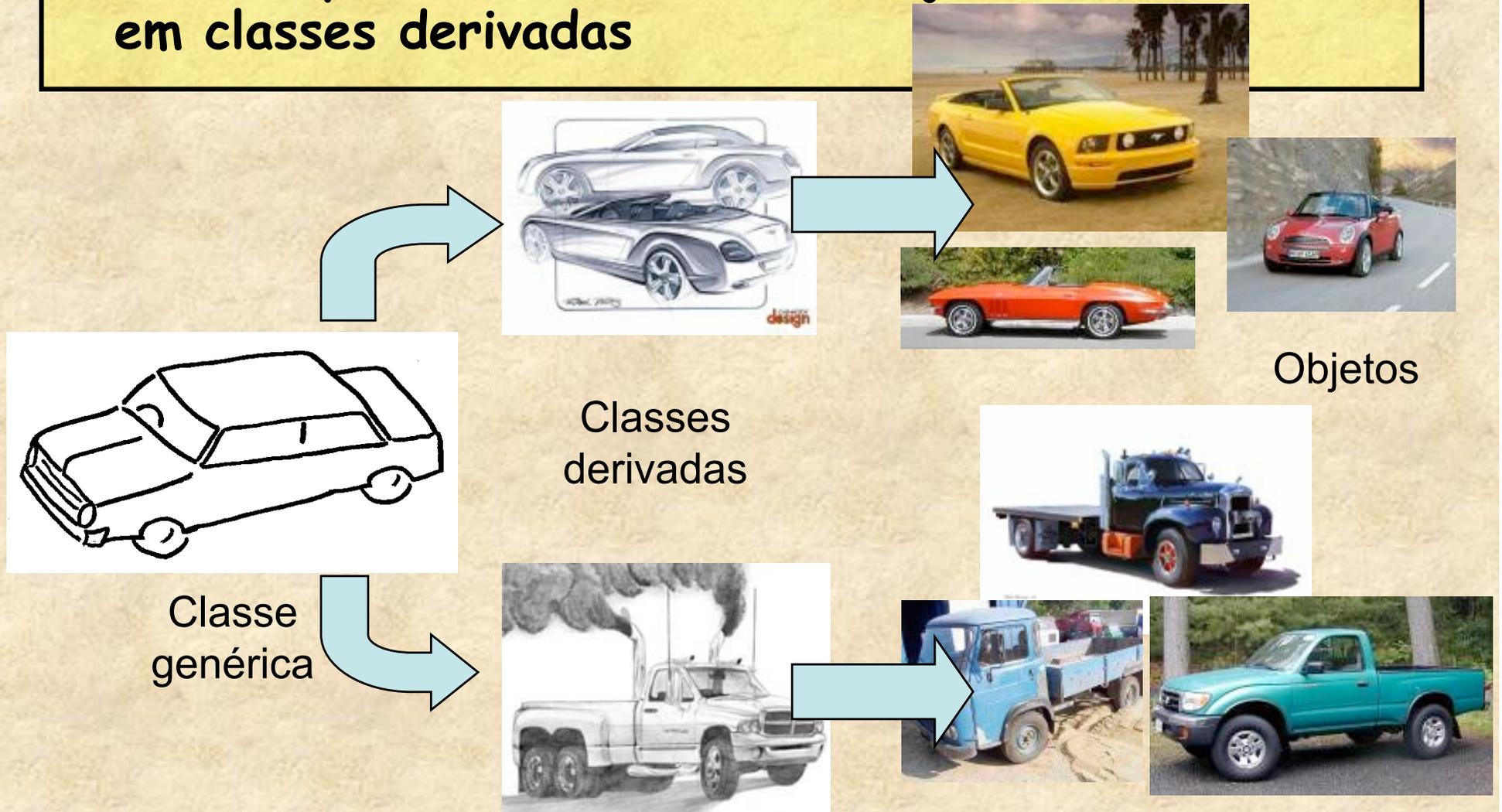
# Classes e objetos em c++

- Classes são os moldes para criação de um objeto em c++
- Objetos são entidades concretas (espaço em memória) criados e organizados com estrutura definida pelas classes



# Classes podem ser derivadas de outras classes

- Em c++ há o conceito de derivação onde classes mais genéricas são criadas e funções/atributos muito específicos de um sub-conjunto são definidas em classes derivadas



# Na prática - definindo uma classe de objetos

Define uma classe chamada Bla

Esta classe é derivada da classe Blu (caso necessário)

```
class Bla : public Blu
{
    protected:
        <membros protegidos>;
    private:
        <membros privados>;
    public:
        <membros publicos>;

    Bla();
    virtual ~Bla();
};
```

Os membros e funções definidas serão tratados como protegidos

Os membros e funções definidas serão tratados como privados

Os membros e funções definidas serão tratados como públicos

Construtor da classe.

Destrutor (~) da classe.

As funções construtoras e destrutoras DEVEM ter o mesmo nome da classe

## Membros de uma classe

- Os membros de uma classe podem ser variáveis, funções, ponteiros ou até mesmo objetos de outras classes
  - Dependem da tarefa a ser executada
- Membros podem ser protected, private ou public
  - Public
    - Não há restrições no acesso a estes membros, sejam em classes derivadas ou pelo usuário
  - Protected
    - Só podem ser acessados pela própria classe ou classes derivadas
  - Private
    - Só podem ser acessados pela própria classe
- Existe um tipo de classe chamada "friend" que pode acessar todos os membros de outra classe mas o seu uso não é recomendado.

# Acessando e criando objetos

- Para saber os métodos (funções) acessíveis de um objeto deve-se olhar a sua definição (classe). Os métodos públicos são acessíveis
  - Os métodos com o mesmo nome da classe são denominados construtores e são utilizados para criar o objeto

```
class TGraph : public TNamed, public TAttLine, public TAttFill,  
              public TAttMarker  
{  
    public:  
        TGraph();  
        TGraph(Int_t n);  
        TGraph(Int_t n, const Int_t* x, const Int_t* y);  
        TGraph(Int_t n, const Float_t* x, const Float_t* y);  
        TGraph(Int_t n, const Double_t* x, const Double_t* y);  
        Double_t GetErrorX(Int_t bin);  
        Double_t GetErrorY(Int_t bin);  
        Double_t GetCovariance();  
        void      SetTitle(const char* title = "");  
        ...  
};
```

Classes das quais esse objeto também faz parte. Métodos públicos dessas classes também são acessíveis

construtores

Métodos diversos

# Criando e destruindo objetos

- Criando objetos no stack

```
void exemplo_obj_1()  
{  
    TH1F h("hist", "histograma", 100, 0, 10);  
    h.SetLineColor(1);  
    h.Draw();  
}
```

Parâmetros  
para criação do  
objeto

Construtor do  
objeto

- O objeto `h` deixa de existir quando a função termina

- Criando objetos no heap (`new` e `delete`)

```
void exemplo_obj_2()  
{  
    TH1F* h = new TH1F("hist", "histograma", 100, 0, 10);  
    h->SetLineColor(1);  
    h->Draw();  
}
```

- Objetos no heap são acessados com ponteiros

- O objeto `h` só deixa de existir com o `delete h`;

# Gráficos e histogramas no ROOT

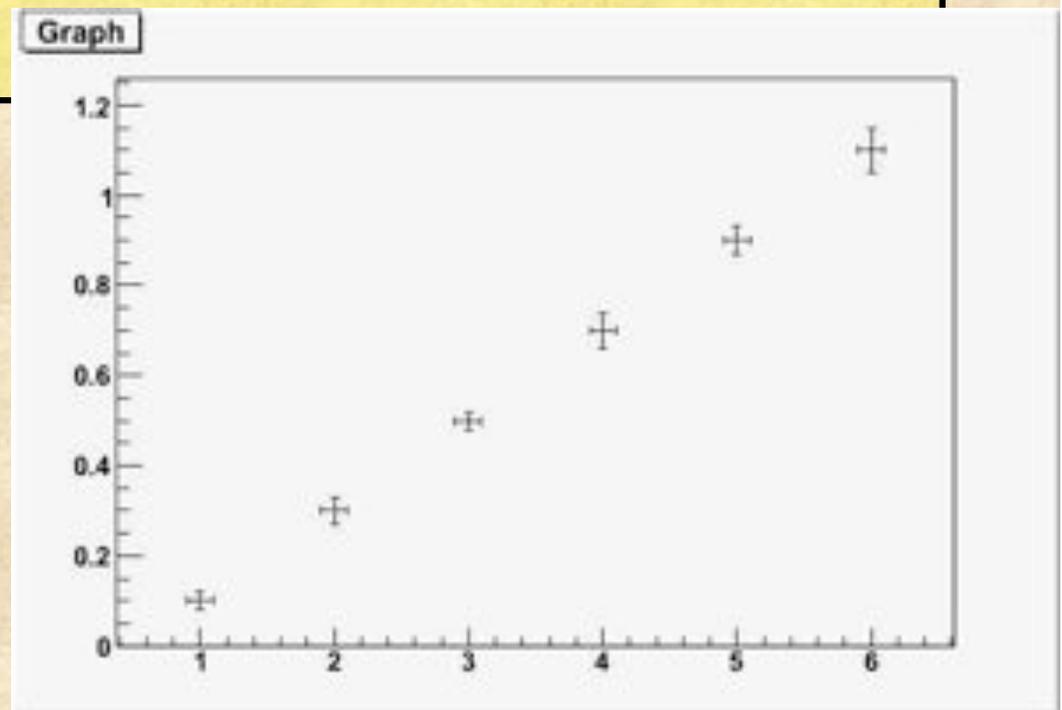
- **Gráficos e histogramas**
  - O ROOT possui uma quantidade enorme de classes para tratar objetos gráficos
  - **Gráficos**
    - TGraph - Gráficos de X e Y simples
    - TGraphErrors - Gráficos com barras de erro
  - **Funções**
    - TF1 - Função de 1 variável  $F=F(x)$
    - TF2 - Função em 2 variáveis  $F=F(x,y)$
  - **Histogramas**
    - TH1 - Histogramas de 1 dimensão
      - TH1I, TH1S, TH1F, TH1D, ... (estabelece a precisão do eixo)
    - TH2 - Histogramas de 2 dimensões
    - TH3 - Histogramas de 3 dimensões

## Gráficos X-Y

- Criar gráficos a partir de uma tabela é como tirar doce da mão de criança...
- TGraph e TGraphError
  - ... = new TGraph(N,x,y);
  - ... = new TGraphErrors(N,x,y,ex,ey);
- Onde
  - N = número de pontos
  - x, y são ponteiros para os vetores com os dados
  - ex, ey são ponteiros para os vetores com os erros

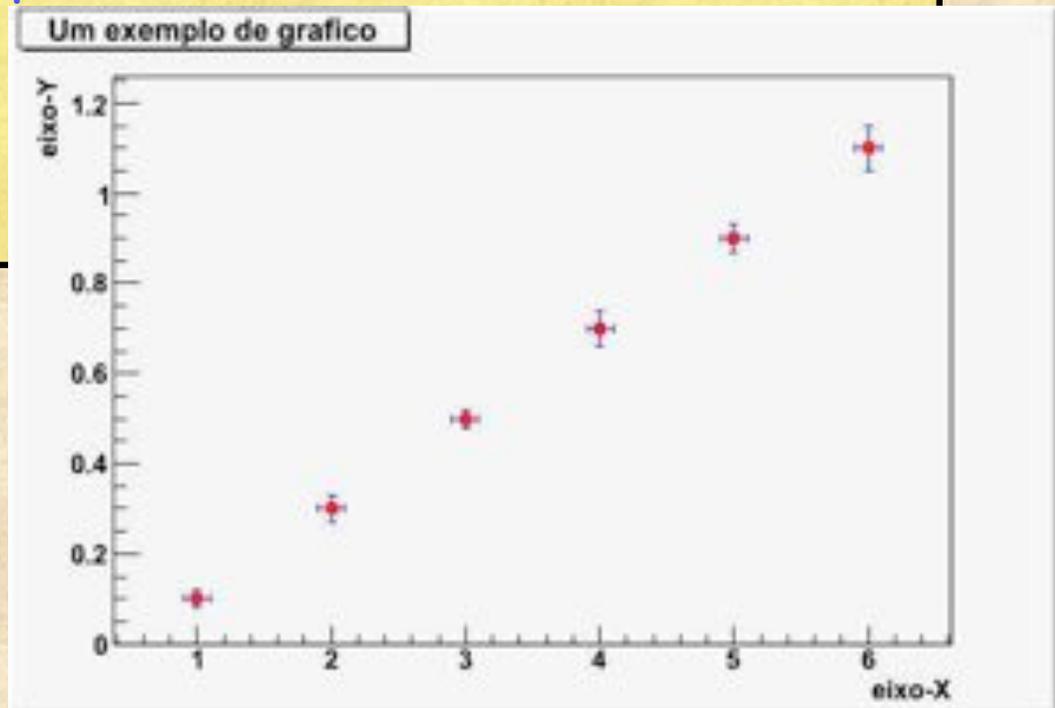
## Um exemplo de gráfico (exemplo\_Tgraph\_1.C)

```
void exemplo_Tgraph_1()  
{  
    float x[] = {1,2,3,4,5,6};  
    float y[] = {0.1,0.3,0.5,0.7,0.9,1.1};  
    float ex[] = {0.1,0.1,0.1,0.1,0.1,0.1};  
    float ey[] = {0.02,0.03,0.02,0.04,0.03,0.05};  
    TGraphErrors *g = new TGraphErrors(6,x,y,ex,ey);  
    g->Draw("AP"); // A desenha os eixos, P desenha pontos  
    return;  
}
```



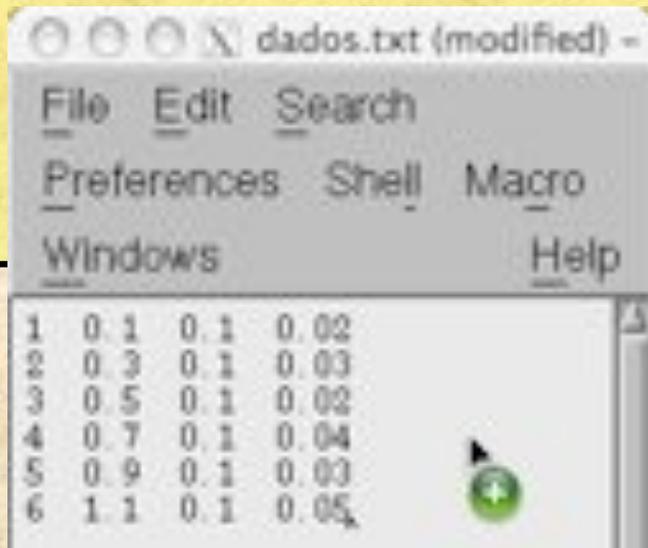
# Um exemplo de gráfico (exemplo\_Tgraph\_2.C)

```
void exemplo_Tgraph_2()  
{  
    float x[] = {1,2,3,4,5,6};  
    float y[] = {0.1,0.3,0.5,0.7,0.9,1.1};  
    float ex[] = {0.1,0.1,0.1,0.1,0.1,0.1};  
    float ey[] = {0.02,0.03,0.02,0.04,0.03,0.05};  
    TGraphErrors *g = new TGraphErrors(6,x,y,ex,ey);  
    g->Draw("AP"); // A desenha os eixos, P desenha pontos  
    g->SetMarkerStyle(20); // estilo do ponto 20 = circulo  
    g->SetMarkerColor(2);  
    g->SetLineColor(4);  
    g->SetTitle("Um exemplo de grafico");  
    g->GetXaxis()->SetTitle("eixo-X");  
    g->GetYaxis()->SetTitle("eixo-Y");  
    return;  
}
```



# Um exemplo de gráfico

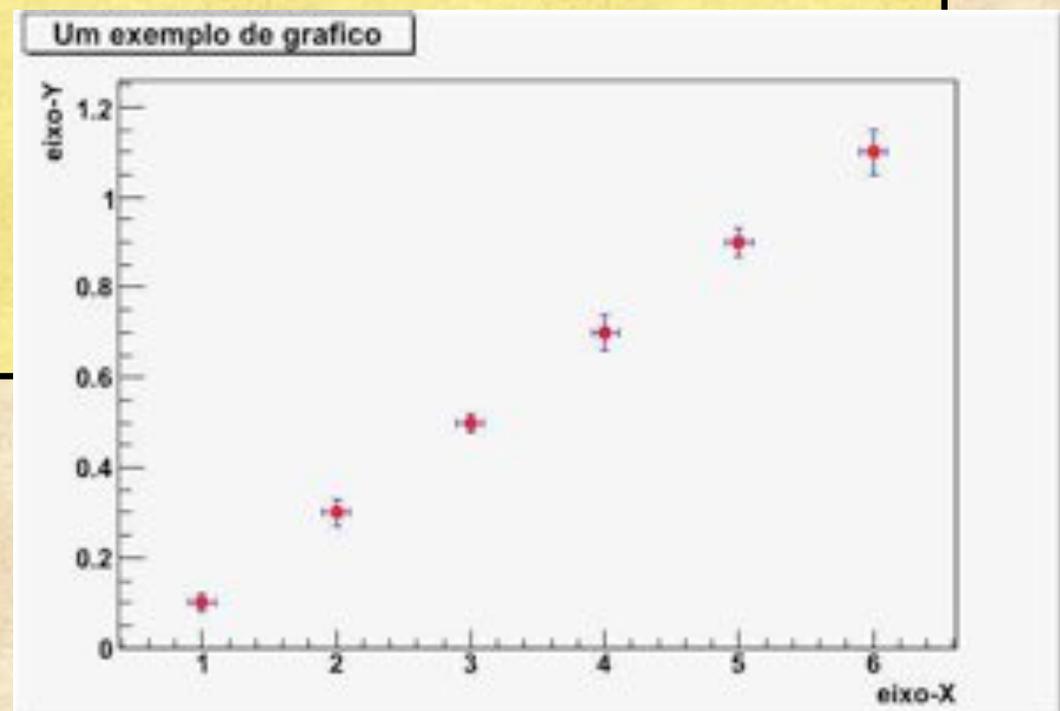
```
void exemplo_TGraph_3()  
{  
  TGraphErrors *g = new TGraphErrors("dados.txt", "%lg %lg %lg %lg");  
  g->Draw("AP"); // A desenha os eixos, P desenha pontos  
  g->SetMarkerStyle(20); // estilo do ponto 20 = circulo  
  g->SetMarkerColor(2);  
  g->SetLineColor(4);  
  g->SetTitle("Um exemplo de grafico");  
  g->GetXaxis()->SetTitle("eixo-X");  
  g->GetYaxis()->SetTitle("eixo-Y");  
  return;  
}
```



dados.txt (modified) -

File Edit Search  
Preferences Shell Macro  
Windows Help

1	0.1	0.1	0.02
2	0.3	0.1	0.03
3	0.5	0.1	0.02
4	0.7	0.1	0.04
5	0.9	0.1	0.03
6	1.1	0.1	0.05



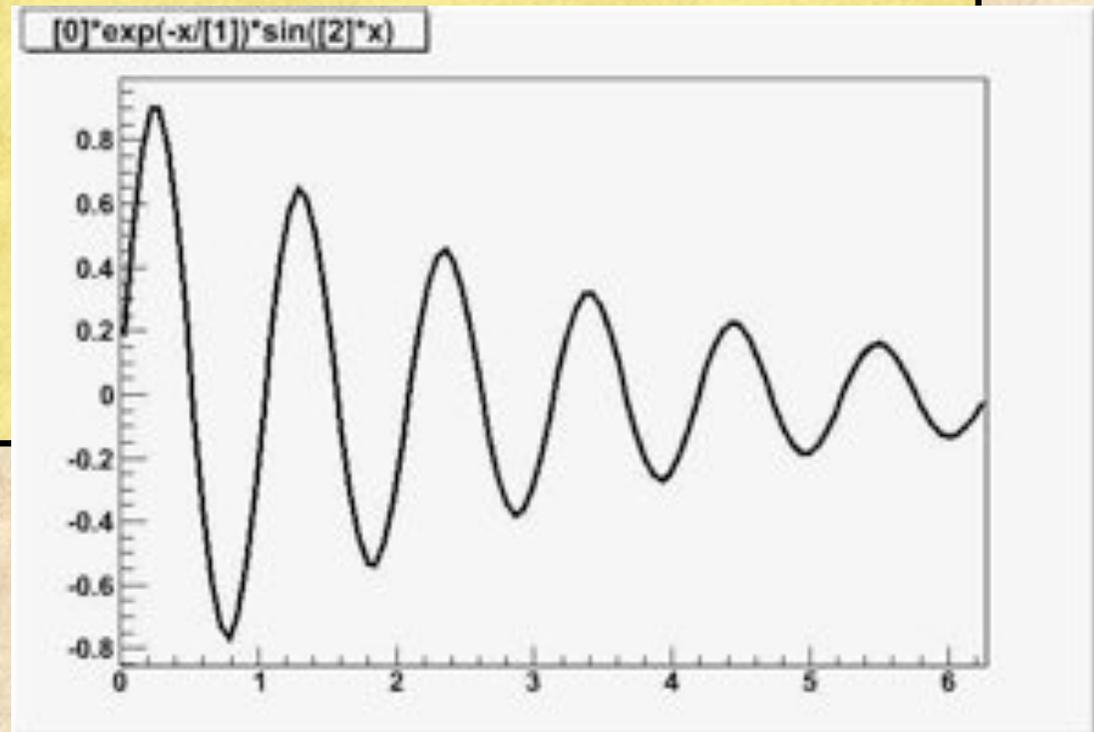
## Criando funções

- O ROOT possui classes para definir funções.
  - TF1, TF2 e TF3
- Uso
  - `TF1 *f = new TF1("nome","formula",min,max);`
- A fórmula deve ser escrita usando a sintaxe padrão de c++.
  - Parâmetros variáveis devem vir entre brackets
    - [0], [1], etc
  - As variáveis são x, y e z
- Alguns métodos interessantes
  - `SetParameter()`, `GetParameter()`, `GetParError()`, `GetChisquare()`, `Eval()`, etc.

## Um exemplo simples

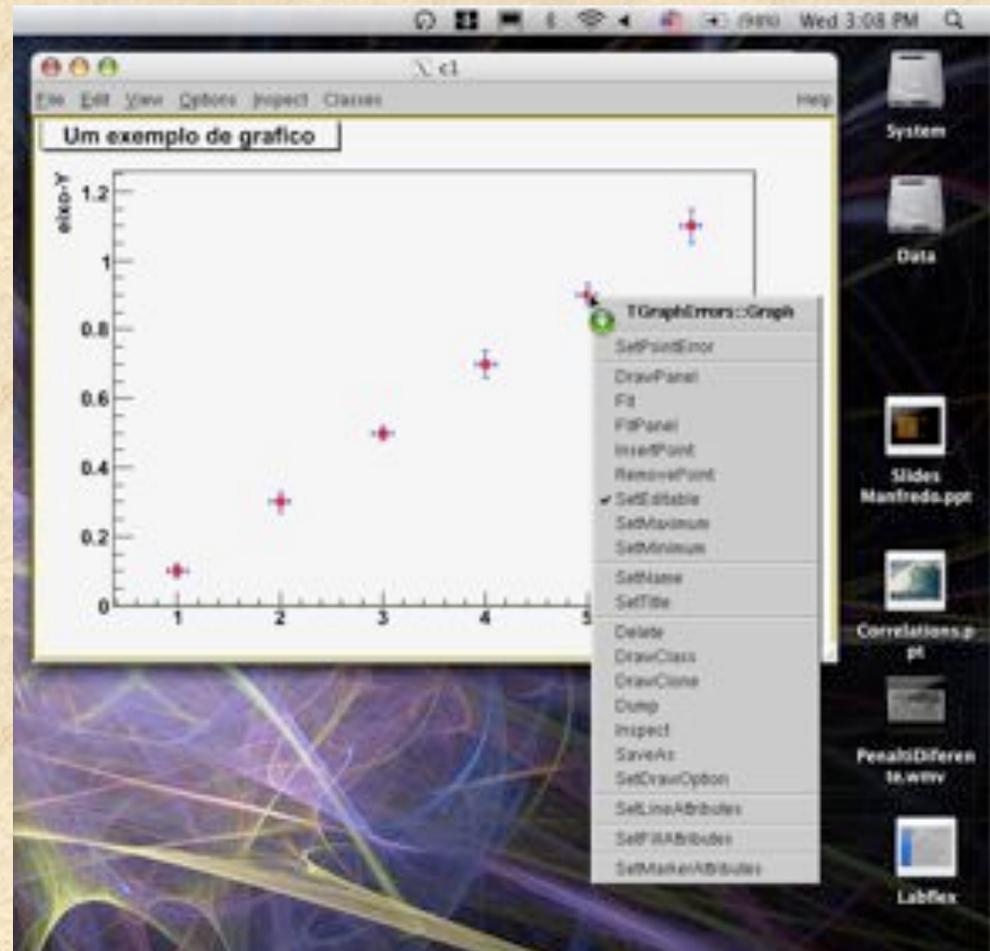
```
void exemplo_func()  
{  
    TF1 *f1 = new TF1("func",  
                      "[0]*exp(-x/[1])*sin([2]*x)",  
                      0,6.28);  
  
    f1->SetParameter(0,1);  
    f1->SetParameter(1,3);  
    f1->SetParameter(2,6);  
    f1->Draw();  
}
```

O fim de um comando só ocorre quando se coloca  
o ;



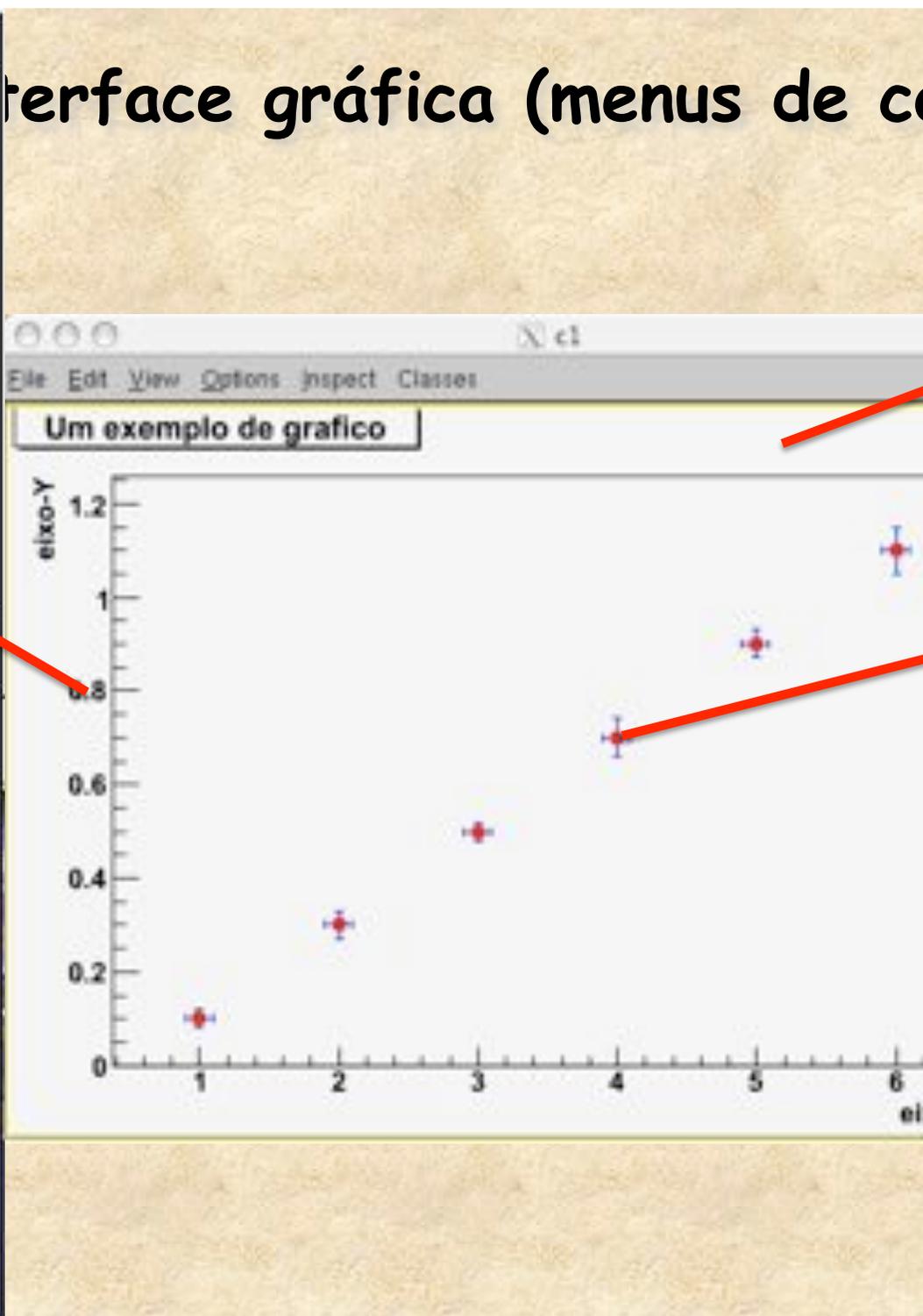
# Outra forma de interface: Mouse

- Sempre que passamos com o mouse sobre elementos gráficos (eixos, pontos de um gráfico, fundo da tela), podemos clicar com o botão direito e abrir uma menu de atributos



# Interface gráfica (menus de co

- TAxis::yaxis
  - CenterLabels
  - CenterTitle
  - LabelsOption
  - RotateTitle
  - SetMoreLogLabels
  - SetNoExponent
  - SetDecimals
  - SetRange
  - SetRangeUser
  - SetTicks
  - SetTimeDisplay
  - SetTimeFormat
  - UnZoom
  - SetName
  - SetTitle
  - DrawClass
  - Dump
  - Inspect
  - SaveAs
  - SetNdivisions
  - SetAxisColor
  - SetLabelColor
  - SetLabelFont
  - SetLabelOffset
  - SetLabelSize
  - SetTickLength
  - SetTitleOffset
  - SetTitleSize
  - SetTitleColor
  - SetTitleFont



- TCanvas::c1
  - DrawClonePad
  - SetGrayscale
  - SetCanvasSize
  - BuildLegend
  - Divide
  - UseCurrentStyle
  - Range
  - SaveAs
  - SetBorderMode
  - SetBorderSize
  - SetCrosshair
  - SetEditable
  - SetFixedAspectRatio
  - SetGridx
  - SetGridy
  - SetLogx
  - SetLogy
  - SetLogz
  - SetName
  - SetTickx
  - SetTicky
  - DrawClass
  - DrawClone
  - Dump
  - Inspect
  - SetLineAttributes
  - SetFillAttributes

# Interface gráfica (menús de contexto)

The image displays a graphical user interface for a graphing application. On the left, a window titled "Um exemplo de grafico" shows a scatter plot with data points and error bars. The x-axis is labeled "eixo-X" and the y-axis is labeled "eixo-Y". A context menu is open over the graph, listing various actions such as "SetPointError", "DrawPanel", "Fit", "FitPanel", "InsertPoint", "RemovePoint", "SetEditable", "SetMaximum", "SetMinimum", "SetName", "SetTitle", "Delete", "DrawClass", "DrawClone", "Dump", "Inspect", "SaveAs", "SetDrawOption", "SetLineAttributes", "SetFillAttributes", and "SetMarkerAttributes". A red arrow points from the "FitPanel" option in the context menu to the "Fit Panel" dialog box on the right. The "Fit Panel" dialog box is titled "Fit Panel" and shows the current selection as "Graph: TGraphErrors". It has two tabs: "General" and "Minimization". The "General" tab is active, showing a "Function" section with a dropdown menu set to "gaus" and a "Set Parameters..." button. The "Fit Settings" section includes a "Method" dropdown set to "Chi-square" and a "Robust" input field set to "1.00". There are several checkboxes for "Fit Options" and "Draw Options". The "Fit" button is highlighted at the bottom of the dialog box.

Um exemplo de grafico

eixo-Y

eixo-X

TGraphErrors::Graph

- SetPointError
- DrawPanel
- Fit
- FitPanel
- InsertPoint
- RemovePoint
- SetEditable
- SetMaximum
- SetMinimum
- SetName
- SetTitle
- Delete
- DrawClass
- DrawClone
- Dump
- Inspect
- SaveAs
- SetDrawOption
- SetLineAttributes
- SetFillAttributes
- SetMarkerAttributes

Fit Panel

Current selection: Graph: TGraphErrors

General | Minimization

Function

Predefined: gaus

Operation:  Nop  Add  Copy

Selected: gaus

Set Parameters...

Fit Settings

Method: Chi-square

Linear fit

Robust: 1.00

No Chi-square

Fit Options

- Integral
- Best errors
- All weights = 1
- Empty bins, weights=1
- Use range
- Improve fit results
- Add to list

Draw Options

- SAME
- No drawing
- Do not store/draw

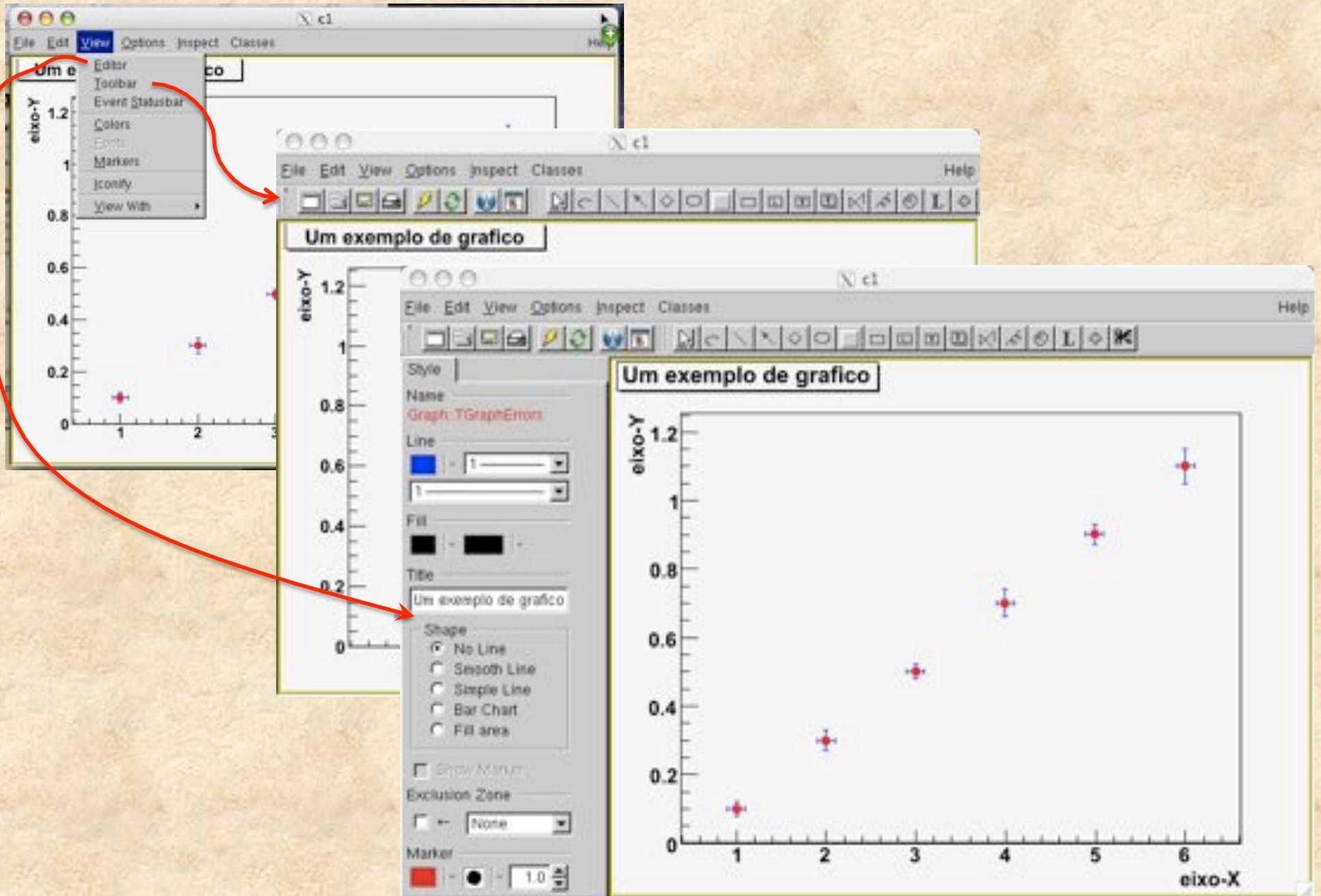
Fit

Reset

Close

LIB Minuit MIGRAD Iter: 5000 Pm DEF

# Interface gráfica (barras de ferramentas)



# Salvando como .C

- Salvando o arquivo no formato ROOT macro
  - File -> Save as
- O ROOT gera um programa em C que, se executado, cria exatamente a mesma figura
  - Muito útil para aprender as manhas

```
exemplo_TGraph_salvo.C - /Volumes/Data/Users/suaide/
File Edit Search Preferences Shell Macro Windows Help
21 gpx->SetPointError(2, 0.1, 0.02);
22 gpx->SetPoint(3, 1.591597, 0.7813525);
23 gpx->SetPointError(3, 0.1, 0.04);
24 gpx->SetPoint(4, 4.387833, 0.3823883);
25 gpx->SetPointError(4, 0.1, 0.03);
26 gpx->SetPoint(5, 6, 1.1);
27 gpx->SetPointError(5, 0.1, 0.05);
28
29 TCI *graph12 = new TCI("graph12", "um exemplo de grafico", 100, 0.38, 6.62)
30 graph12->SetWindow(0);
31 graph12->SetXname(1.257);
32 graph12->SetYname(0);
33 graph12->SetState(0);
34 graph12->GetXaxis()->SetTitle("x=x");
35 graph12->GetYaxis()->SetTitle("y=y");
36 gpx->SetHistogram(graph12);
37
38 gpx->Draw("ap");
39
40 TPad *p4 = new TPad(0.01, 0.340647, 0.381265, 0.995, "blinc");
41 p4->SetName("title");
42 p4->SetBorderSize(2);
43 p4->SetFillColor(13);
44 TText *text = p4->AddText("um exemplo de grafico");
45 p4->Draw();
46
47 TEllipse *ellipse = new TEllipse(1.817259, 0.328784, 0.4751263, 0.1291917,
48 ellipse->Draw();
49 TPad *pave = new TPad(2.558345, 1.013567, 3.08394, 1.2151, 4, "br");
50 pave->Draw();
51 TCurve *curve = new TCurve(2.218198, 0.7891167, 4.483243, 0.415
52 curve->Draw();
53 TArrow *arrow = new TArrow(5.314721, 0.4155083, 5.037563, 0.8513667, 0.05, "+"
54 arrow->SetFillColor(3);
55 arrow->SetFillStyle(1001);
56 arrow->Draw();
57
58 p4 = new TPad(2.838503, 0.1291917, 6.436546, 0.321283, "br");
59 p4->SetFillColor(13);
60 TLine *line = p4->AddLine(0, -0.6727272, 8, -0.6727272);
61 text = p4->AddText("testando a barra de ferramentas");
62 p4->Draw();
63 c1->Modified();
64 c1->cd();
65 c1->SetSelected(c1);
66 c1->ToggleToolBar();
```

## Como obter informações

- Vários tutoriais de ROOT em
  - <http://www.dfn.if.usp.br/~suaide/>
- Referências e documentação do ROOT
  - <http://root.cern.ch>
    - Página principal do root
  - <http://root.cern.ch/root/Categories.html>
    - Documentação sobre as classes do root
  - <http://root.cern.ch/root/Tutorials.html>
    - Tutoriais com exemplos simples, passo a passo
  - <http://root.cern.ch/root/HowTo.html>
    - Como resolver problemas comuns