Parte I
Istogrammi

Come costruire un istogramma:

- Come **Puntatore**:

```
TH1F *h1 = new TH1F("h1", "Titolo", nBins, x lowest bin, x highest bin);
```

- Come **Istanza**:

```
TH1F h1 = TH1F("h1", "Titolo", nBins, x lowest bin, x highest bin);
```

Per riempire ciascuna occorrenza della variabile x:

```
h1 -> Fill(x);
h1.Fill(x);
```

Il metodo Fill ha anche un altro argomento ed e' il peso di quell'ingresso:

```
h1 -> Fill(x, 8);
```

Per disegnare un istogramma:

```
h1 -> Draw();
h1.Draw();
```

Sono disponibili altre varie opzioni:

```
h1 -> Draw("E"); // Visualizza gli errori
h1 -> SetMarkerStyle(22); //Sclego il tipo di simbolo
h1 -> Draw("E,P"); //Con errori e simbolo
h1 -> Draw("HIST,SAME"); //Sovrappongo stesso istogramma con linea continua
h1 -> GetMean(); //Restituisce la media
h1 -> GetRMS(); //Radice della varianza
h1 -> GetMaximum(); //Contenuto massimo dei bin
h1 -> GetMaximumBin(); //locazione del massimo
h1 -> GetBinContent(0); //Ritorna il numero di UNDERFLOW
h1 -> GetBinContent(Nbin + 1); //OVERFLOW
h1 -> GetBinError(ibin); //Errore contenuto nel bin
h1 -> SetBinContent(ibin, val); //assegna al contenuto del bin "ibin" il contenuto "
val"
h1 -> GetEntries(); //Numero totale di conteggi
h1 -> Integral(ibin1, ibin2); //integrale nel range
h1 -> GetIntegral(); //Array dei conteggi cumulativi
h1 -> GetMeanError();
h1 -> GetRMSError();
```

Esempio 1

```
void histo(Int_t nev = 1E3, Float_t mean = 0., Float_t width = 1.){
    TH1F * h = new TH1F("h", "␣x␣distribution", 100, -5., 5);
    Float_t x = 0;

    for(Int_t i = 0; i < nev; i++){
        x = gRandom -> Gaus(mean, width); // un metodo di generazione Monte Carlo
secondo una pdf gaussiana
        h -> Fill(x);
    }

    h -> Draw("E"); //Con errori sugli ingressi nei bin
    h -> Draw("HISTO,SAME"); //sovrapporre anche come linea continua

    TFile *file = new TFile("example.root", "RECREATE"); //aprire il file ROOT
    h -> Write(); //scrivere su file ROOT
    file -> Close(); //chiudere il file ROOT
}
```

Root Global Variables

- **gROOT**: Informazione globale relativa alla sessione corrente attraverso il quale si puo accedere praticamente a qualunque oggetto creato durante la sessione di ROOT
- **gFile**: puntatore al root file corrente
- **gStyle**: puntatore alla funzionelita per gestire lo stile grafico
- **gPad**: puntatore alla pad corrente
- **gRandom**: puntatore al generatore di numeri random

Esempio 2

```
//General settings for graphics
void setStyle(){
    gROOT -> SetStyle("Plain");
    gStyle -> SetPalette(57);
    gStyle -> SetOptTitle(0);
}

void maxwell() {
// funzione principale della macro

//creazione istogramma
    TH1F *h1 = new TH1F("h1", "TempiCaduta", 8, -0.5, 15.5);

//lettura da file ascii e riempimento istogramma
    ifstream in;
    in.open("maxwell.dat");
    Float_t x, y;
    while (1) {
```

```
            in >> x >> y;
            if ( ! in.good() ) break;
            h1 -> Fill(y);
        }
        in.close();

    //Canvas, la finestra grafica....
        TCanvas *cMaxwell = new TCanvas("cMaxwell", "TempiDiCadutaDelPendoloDiMaxwell");
    //cosmetica: assi, colore, spessore linea,tipo di Marker...
        h1 -> GetXaxis() -> SetTitle("Tempi di Caduta (s)");
        h1 -> GetYaxis() -> SetTitleOffset(1.3);
        h1 -> GetYaxis() -> SetTitle("Occorrenze");
        h1 -> SetFillColor(kBlue); //(KBlue=2)
        h1 -> SetMarkerStyle(4); //Open Circle
        h1 -> SetLineWidth(2); //spessore linea
        gStyle -> SetOptStat(112210); //no nome, entries, media e rms con errori, under (
    over)flow
        h1 -> Draw();
        h1 -> Draw("E,same");

        cout << "*------------------------------------------------------------*" <<endl;

        cout << "OccorrenzeTotali:" << h1 -> GetEntries() << endl;
        cout << "MediaIstogramma:" << h1 -> GetMean() << "+/-" << h1 -> GetMeanError() <<
    endl;
        cout << "RMSIstogramma:" << h1 -> GetRMS() << "+/-" << h1 -> GetRMSError() << endl;
        cMaxwell -> Print("cMaxwell.pdf");
        cMaxwell -> Print("cMaxwell.C");
        cMaxwell -> Print("cMaxwell.root");
    }
```

Operazioni tra Istogrammi

Operazioni su istogrammi omologhi (stesso numero di bin e stasso range). Si puo fare attraverso overload dell'operatore somma, moltiplicazione, ...

```
        TH1F h1;
        TH1F h2 = 3 * h1;
        TH2F h3 = h1 + h2;
        // Questa modalita funzione solo per le istanze e non per i puntatori

        //OPPURE ATTRAVERSO I METODI DELLA CLASSE

        h1 -> Add(...) //somma
        h1 -> Multiply(...) //moltiplicazione
        h1 -> Divide(...) //divisione
```

Esempio 3

```
        //... creazione di due istogrammi
```

```
TH1F *h1 = new TH1F (...);
TH1F *h2 = new TH1F (...); //omologhi

// cicli di lettura e riempimento istogrammi

//dopo averli riempiti confrontiamo i dati facendo il rapporto

TH1F *hRatio = new TH1F(*h1);
hRatio -> Divide(h1, h2, 1, 1);

//sommiamo i due campioni in un unico istogramma

TH1F *hSum = new TH1F (*h1);
hSum -> Add(h1, h2, 1, 1);

//...
```

Parte II
Grafici

Abbiamo due tipi di classi per creare grafici:

- **TGraph**: rappresenta una serie di N coppie di due quantita X, Y (ad esempio variabili fisiche)
- **TGraphErrors**: (classe derivata) consente di includere anche gli errori EX, EY sulle quantita X, Y

classe TGraph, Due costruttori principali:

```
TGraph (Int_t n, const Double_t *x, const Double_t *y)

//n e il numero di punti
//x e y sono i nomi degli array di punti x e y

TGraph(const char *filename, const char *format = "%lg%lg", Option_t *option = "")

//Il file di input deve contnere almeno due colonne di numeri
//Il formato della stringae di default "%lg%lg" ( 2 double )
//Opzioni per skippare colonne ( "%lg%*lg%lg" ) e interpretare delimitatori differenti
dal blank

TGraphErrors(Int_t n, const Double_t *x, const Double_t *y, const Double_t *ex = 0,
const Double_t *ey = 0)

TGraphErrors(const char *filename, const char *format = "%lg%lg%lg%lg", Option_t *option
 = "")

//il file di input deve  contenere almeno tre colonne di numeri
```

Alcuni metodi:

```
garph -> GetPoint(i, x, y) //recupera i contenuti dell'i-esimo punto del grafico
```

```
graph -> GetX()
graph -> GetY()
graph -> GetN()

graph -> Integal()

graph -> GetCorrelationFactor()
graph -> GetCovariance()

//Fare un fit del grafico

graph -> Fit(const char *formula)
graph -> Fit (TF1 *f1) // funzione precedentemente definita
```

Parte III
Funzioni

Classe per funzioni di una variabile TF1

- consente di rappresentare un'espressione C++ in cui la variabile è x
- funzioni definite dll'utente
- Sono disponibili anche dei BUILT IN function object (gaus, expo, poln)
- corrispondenti in 2, 3, dimensioni TF2, TF3

Primo metodo per definire una funzione:

```
TF1 *f1 = new TF1 ("f1", "sin(x) / x", 0, 10)
f1 -> Draw()
```

Secondo metodo per definire una funzione:

```
TF1 *f1 = new TF1("f1", "[0] * x * sin([1]x)", -3, 3)

f1 -> SetParameter(0,10)
f1 -> SetParameter(1, 5)

f1 -> Draw()
```

Terzo metodo (più generale):

```
Double_t MyFunction (Double_t *x, Double_t *par){
    Float_t xx = x[0];
    Double_t val = TMath::Abs(par[0] * sin(par[1] * xx) / xx);
}

TF1 *f1 = new TF1 ("f1", MyFunction, 0, 10, 2); // 2 numero di parametri in MyFunction
f1 -> SetParameters(2, 1); // Inizializza i due parametri a 2 e 1
```

Alcuni metodi utili:

```
f -> Eval(x) // valutare la funzione in un punto
f -> Integral(a, b)
f -> DrawDerivative()
f -> DrawIntegral() // funzione cumulativa

// FITTING

// Per fare il fit di un istogramma con la funzione scelta

TF1 *fn1 = new TF1("f1", "[0]␣*␣x␣*␣sin([1]␣*␣x)", -3, 3)

aHistogram -> Fit("f1")
aHistogram -> Fit(fn1)

aHistogram -> Fit("gaus")
aHistogram -> Fit("expo")
```