Podstawy ROOTa

Maciej Trzebiński

Instytut Fizyki Jądrowej
Polskiej Akademii Nauk

Praktyki studenckie na LHC

IFJ PAN

6 lipca 2015

W wyniku zderzenia cząstek na akceleratorze mogą powstać inne cząstki. Pojedyncze zderzenie nazywamy **przypadkiem**.
Przykład: W akceleratorze LHC krążą w przeciwnych kierunkach dwie paczki po 10^{11} protonów. Załóżmy, że w pewnym miejscu paczki się mijają tak, że dokładnie jedna para protonów oddziałuje (zderza się ze sobą). W wyniku tego zderzenia powstaje 10 pionów dodatnich (π^+), 9 pionów ujemnych (π^-), neutron, proton i 12 fotonów. Każda z tych cząstek może zostać zmierzona w detektorze, np. możemy wyznaczyć jej trajektorię, pęd, energię *itd*.
Wiązki krążą nadal i po kilku "minięciach" znów następuje sytuacja, w której dokładnie dwa protony się zderzają. Tym razem produkują się dwa protony, trzy piony dodatnie i trzy piony ujemne.
Proszę zauważyć, że:

- te zderzenia są niezależne – jedno nie ma wpływu na drugie,
- ilość informacji dostępnych w danym zderzeniu jest różna (w pierwszym zderzeniu informację o 32 pędach, w drugim o 8),
- rodzaj informacji jest ten sam – tzn. możemy zdefiniować uniwersalne zmienne opisujące np. ilość cząstek w danym przypadku (n = 32 w pierwszym oraz n = 8 w drugim), pęd każdej cząstki (np. px_1, px_2, ..., px_n).

Makro-świat

- Determinizm
- Jeśli znamy warunki początkowe (położenia i prędkości) oraz działające siły możemy przewidzieć co się stanie

Mikro-świat

- Badanie mikro-świata – zderzanie cząstek
- Rozmiar protonu: 1 fm $= 10^{-15}$ m $= 10^{-12}$ mm
- Rozmiar elektronu: mniejszy niż 1/1000 protonu (nie wiadomo jak mały)
- Nie da się zmierzyć parametrów początkowych
- Efekty kwantowe – nawet jeśli znalibyśmy warunki początkowe, nie można przewidzieć wyniku

Dlaczego statystyka?

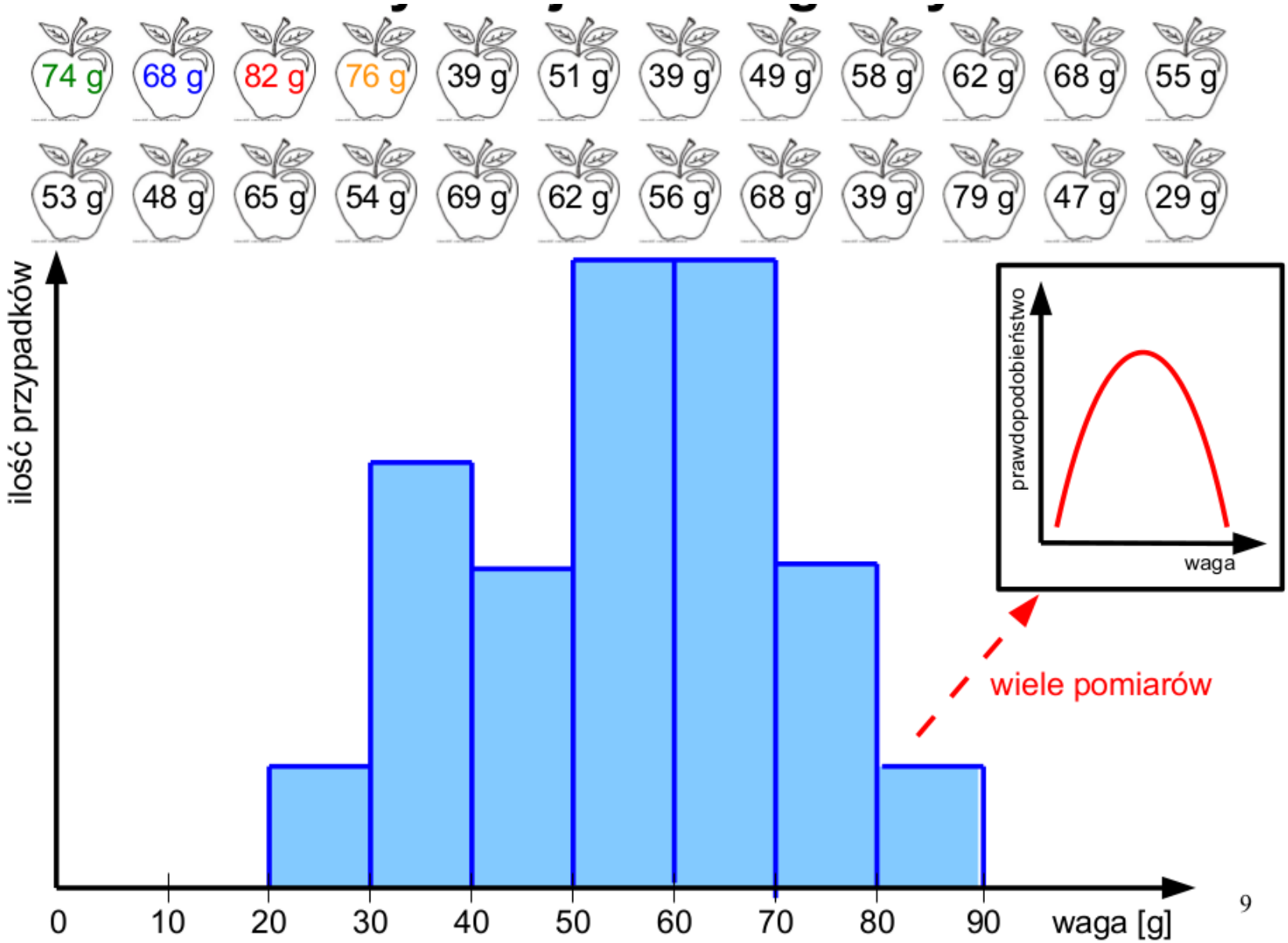
- Nie znamy dokładnie warunków początkowych
- Nawet jeśli znalibyśmy warunki początkowe, nie można przewidzieć wyniku
- Z tego samego eksperymentu dostajemy raz taki wynik, raz inny
- Podstawowe prawa fizyki rządzą prawdopodobieństwami (częstościami występowania) określonych wyników
- Analiza prawdopodobieństw $\longrightarrow$ **statystyka**

Analiza danych – jak odróżnić jabłka od gruszek

?

Cechy charakterystyczne:
- waga
- kolor,
- kształt,
- zapach,
- smak.

Klasyfikacja – histogramy



Analiza danych – jak odróżnić jabłka od gruszek

?

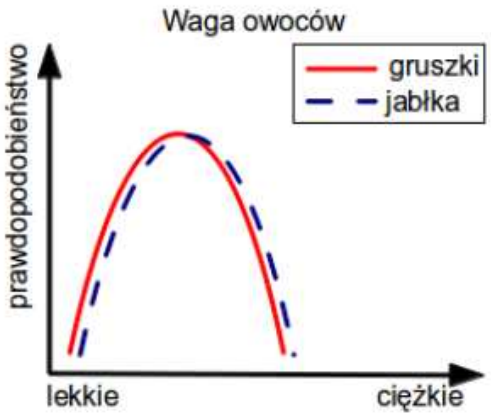
Cechy charakterystyczne:
- waga
- kolor,
- kształt,
- zapach,
- smak.

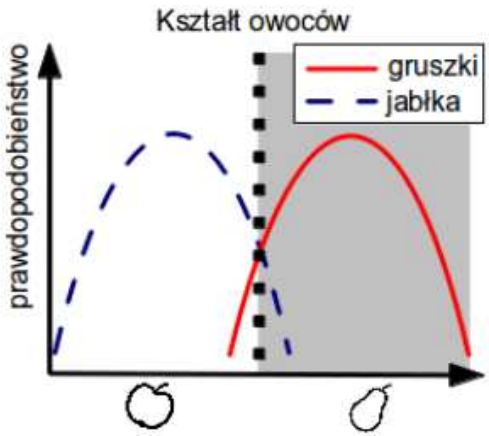
zbliżone cechy

dobra zmienna!

skomplikowana analiza (chemiczna)

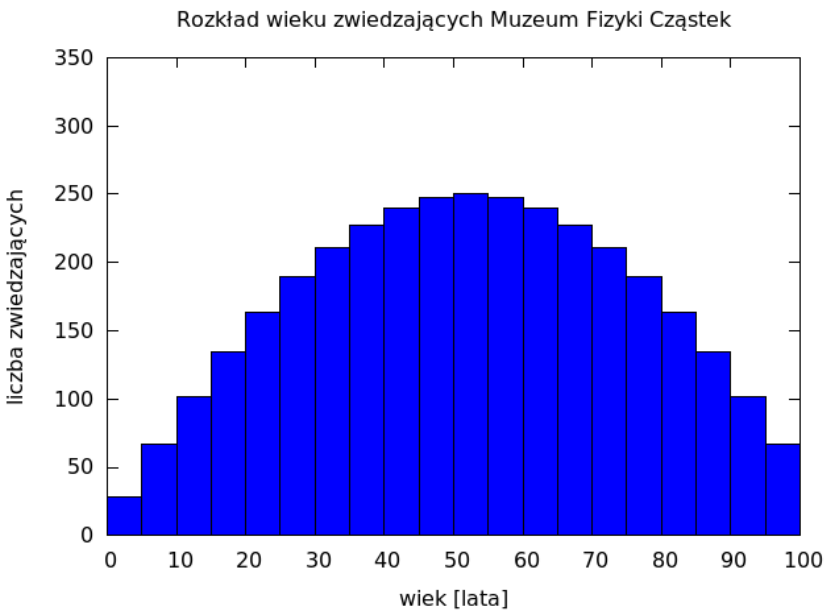
Uwaga!
Zawsze znajdą się gruszki mające wiele cech jabłek!



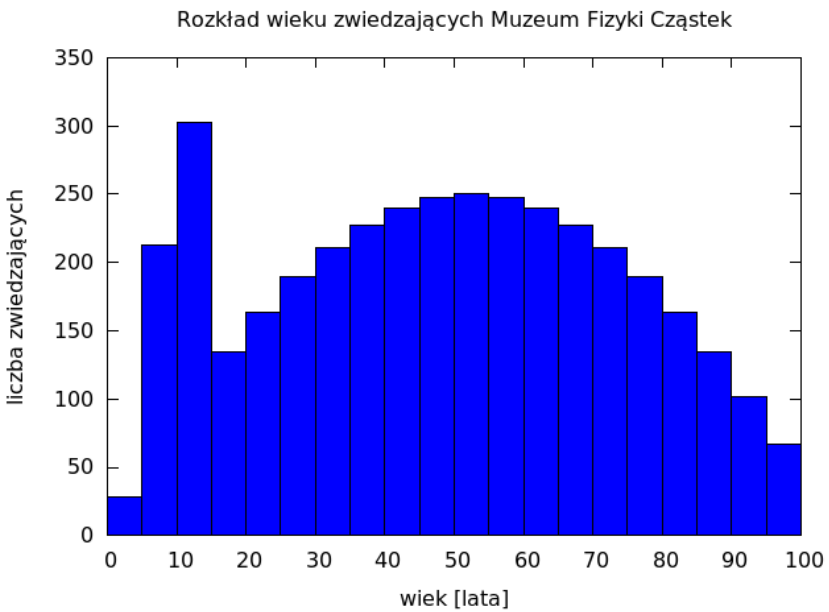


Wiek zwiedzających muzeum Fizyki Cząstek

Typowy rozkład:

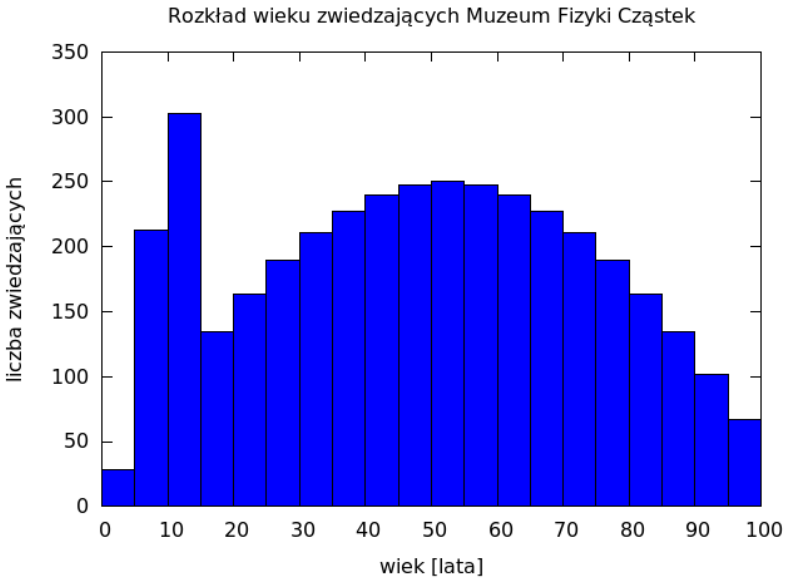


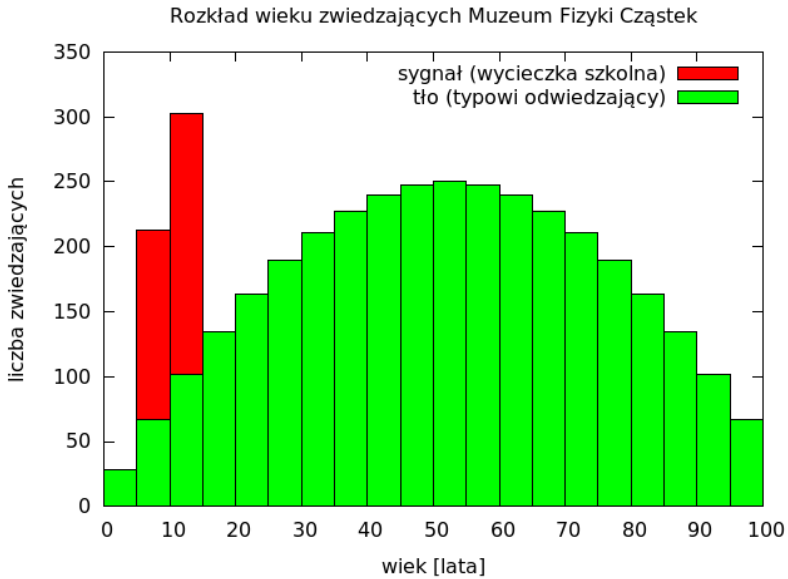
Pewnego razu:

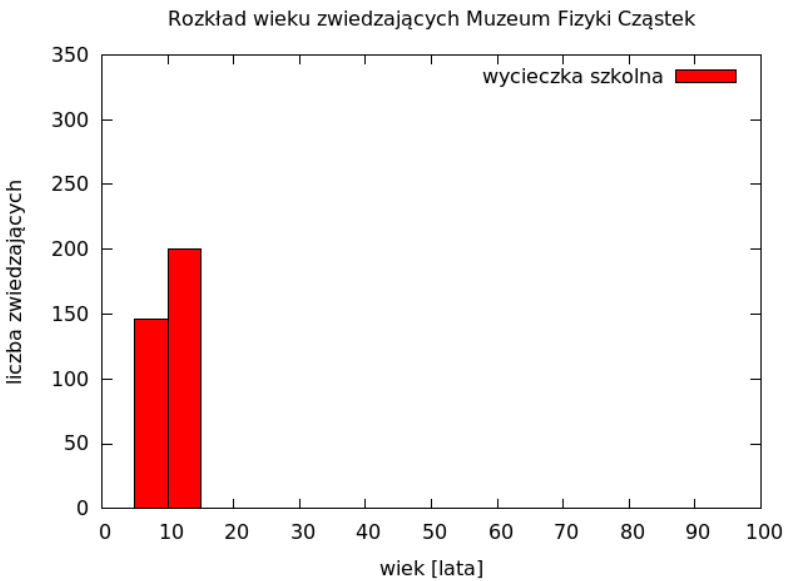


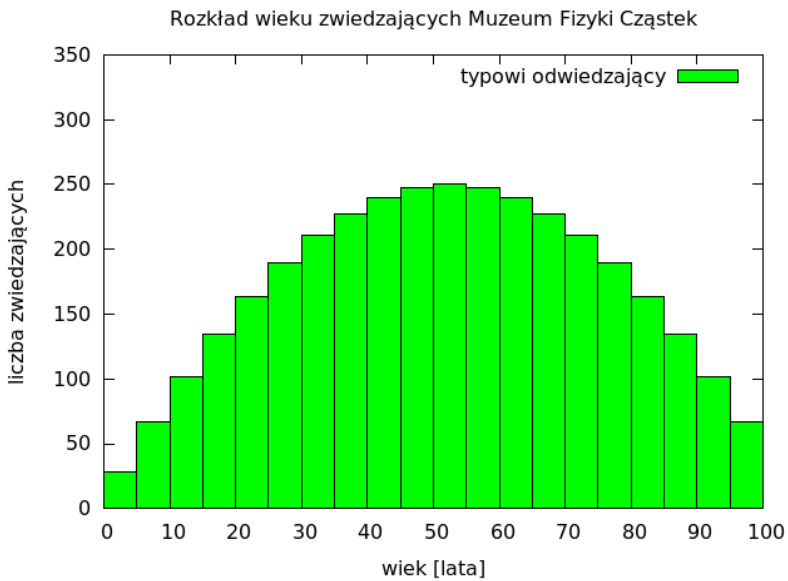
O czym świadczy taka zmiana?

„Sygnał” i „tło”


Rozkład wieku zwiedzających Muzeum Fizyki Cząstek
liczba zwiedzających
wiek [lata]
0
50
100
150
200
250
300
350
0
10
20
30
40
50
60
70
80
90
100


Rozkład wieku zwiedzających Muzeum Fizyki Cząstek
sygnał (wycieczka szkolna)
tło (typowi odwiedzający)
liczba zwiedzających
wiek [lata]
0
50
100
150
200
250
300
350
0
10
20
30
40
50
60
70
80
90
100


Rozkład wieku zwiedzających Muzeum Fizyki Cząstek
wycieczka szkolna
liczba zwiedzających
wiek [lata]
0
50
100
150
200
250
300
350
0
10
20
30
40
50
60
70
80
90
100


Rozkład wieku zwiedzających Muzeum Fizyki Cząstek
typowi odwiedzający
liczba zwiedzających
wiek [lata]
0
50
100
150
200
250
300
350
0
10
20
30
40
50
60
70
80
90
100

Cel oprogramowania ROOT

Stworzenie oprogramowania do analizy danych w fizyce wysokich energii:

- grafika przedstawiająca dane (histogramy, punkty, ...)
- struktury danych (ntuple, drzewa)
- procedury do analizy danych
- pliki skompresowane zawierające dane w formacie niezależnym od systemu operacyjnego

Instalacja:

http://root.cern.ch/drupal/content/downloading-root

Na naszym systemie w CC1 pakiet ROOT został już zainstalowany.

Uruchomić program ROOT:
`root`

Zamknąć program ROOT:
`.q` (w konsoli programu)

Otworzyć program ROOT z opcją "-l", a następnie go zamknąć:
`root -l`
`.q`

Ze strony praktyk ściągnąć przykładowy plik w formacie ROOT (mc_signal.root):
za pomocą przeglądarki (zapisać w katalogu /media/cw_ROOT) lub polecenia
`wget`
`http://atlas.ifj.edu.pl/praktyki/materialy/cwiczenie6/mc_signal.root`

Co jest w pliku?

Otworzyć plik programem ROOT:
`root -l mc_signal.root`

Wylistować zawartość pliku:
`.ls` (w konsoli programu)
Co znaczą wyświetlone informacje?

Przypadki znajdują się w `ntuple`. Wyświetlić informacje o pierwszym przypadku:
`ntuple->Show(1)`
Co zostało wyświetlone?

Wyświetlić informację o piątym przypadku.

Wyświetlić informację o wszystkich przypadkach:
`ntuple->Scan()`
Jakie informacje zostały wyświetlone? Jest tego trochę dużo (10 000), nie będziemy oglądać wszystkich (q)

Wyświetlmy informację o pierwszych 10 przypadkach. Na logikę:
`ntuple->Scan(10)`
Nie działa? Jaki błąd został wyświetlony? Jaka składnia została podpowiedziana przez ROOTa?
`ntuple->Scan(" ", " ", " ", 10)`
`ntuple->Scan(" ", " ", " ", 10, 25)`

`TBrowser a`
lub
`new TBrowser`

Zadania.
Używając interface graficznego:

- wyświetlić zawartość pliku `mc_signal.root`,
- wyświetlić zawartość drzewa `ntuple`,
- wyświetlić histogram zmiennej `m`,
- wyświetlić histogram zmiennej `x1`,
  - ile jest wejść do histogramu?
  - jaka jest liczba binów?
  - w jakim zakresie rysowane są wartości?
  - jaka jest średnia?
- zmienić skalę na osi y na logarytmiczną,
- wyświetlić histogram zmiennej `x6` i zmienić skalę na osi y na liniową,
- czy wartości w histogramach są związane z wartościami wyświetlanymi wcześniej za pomocą poleceń `Show` oraz `Scan`?

Tworzenie klasy

ROOT oferuje nam automatyczne narzędzia do tworzenia klas:
`ntuple->MakeClass("pierwszy_program_ROOT")`

Jaka informacja została podana przez ROOTa?

Wyjść z ROOTa i wylistować zawartość katalogu. Powstały dwa nowe pliki: główny (**.C**) oraz nagłówkowy (**.h**). Pliki te są tzw. makrem programu ROOT – można je uruchomić bezpośrednio przy jego użyciu. Mają składnię opartą na C++.

Uruchomić makro `pierwszy_program_ROOT.C`
`root -l pierwszy_program_ROOT.C`
lub
`root -l`
`.x pierwszy_program_ROOT.C`

Struktura pliku **.h** – nagłówki oraz klasa

```
1  #ifndef pierwszy_program_ROOT_h
2  #define pierwszy_program_ROOT_h
3  #include <TROOT.h>
4  #include <TChain.h>
5  #include <TFile.h>
6  class pierwszy_program_ROOT {
7  public :
8     TTree          *fChain;   //!
9     Int_t           fCurrent; //!
10    // Declaration of leaf types
11    Float_t         m;
12    Float_t         x1;
13    Float_t         x2;
14    // List of branches
15    TBranch        *b_m;   //!
16    TBranch        *b_x1;   //!
17    TBranch        *b_x2;   //!
18
19    pierwszy_program_ROOT(TTree *tree=0);
20    virtual ~pierwszy_program_ROOT();
21    virtual Int_t    Cut(Long64_t entry);
22    virtual Int_t    GetEntry(Long64_t entry);
23    virtual Long64_t LoadTree(Long64_t entry);
24    virtual void     Init(TTree *tree);
25    virtual void     Loop();
26    virtual Bool_t   Notify();
27    virtual void     Show(Long64_t entry = -1);
28 };
29 #endif
```

Struktura pliku **.h** – inicjalizacja gałęzi

```
1  void pierwszy_program_ROOT::Init(TTree *tree)
2  {
3     // Set branch addresses and branch pointers
4     if (!tree) return;
5     fChain = tree;
6     fCurrent = -1;
7     fChain->SetMakeClass(1);
8
9     fChain->SetBranchAddress("m", &m, &b_m);
10    fChain->SetBranchAddress("x1", &x1, &b_x1);
11    fChain->SetBranchAddress("x2", &x2, &b_x2);
12    Notify();
13 }
```

Struktura pliku .C

```
1  #define pierwszy_program_ROOT_cxx
2  #include "pierwszy_program_ROOT.h"
3  #include <TH2.h>
4  #include <TStyle.h>
5  #include <TCanvas.h>
6
7  void pierwszy_program_ROOT::Loop()
8  {
9     if (fChain == 0) return;
10
11    Long64_t nentries = fChain->GetEntriesFast();
12
13    Long64_t nbytes = 0, nb = 0;
14    for (Long64_t jentry=0; jentry<nentries;jentry++) {
15       Long64_t ientry = LoadTree(jentry);
16       if (ientry < 0) break;
17       nb = fChain->GetEntry(jentry);   nbytes += nb;
18       // if (Cut(ientry) < 0) continue;
19    }
20 }
```

Zmodyfikować plik **.C** tak aby:

- wyświetlił na ekran numer przypadku,
- wyświetlił na ekran wartość parametru m dla danego przypadku,
- wyświetlił na ekran wartość parametru m tylko dla przypadków, gdy $x1 > 1$.

Tworzenie histogramów i plików

Definicja histogramu jednowymiarowego (klasa `TH1F.h`):

```
TH1F *h1 = new TH1F("nazwa", "tytul", nbin, xmin, xmax);
```

nbins – liczba binów (int)
xmin – początek osi X (float lub double)
xmax – koniec osi X (float lub double)

Wypełnienie histogramu danymi:

```
h1->Fill(zmienna);
```

Definicja histogramu dwuwymiarowego `TH2F.h`:

```
TH2F *h2 = new TH2F("nazwa", "tytul", nbinx, xmin, xmax, nbiny,
ymin, ymax);
h2->Fill(zmiennax, zmiennay);
```

Tworzenie pliku:

```
TFile * plik = new TFile("plik.root", "RECREATE");
```

Inne opcje to: "NEW" – tworzenie nowego, błąd gdy już istnieje, "READ" – otwarcie tylko do odczytu

Zapisanie histogramu do pliku:

```
h1->Write();
```

Zamknięcie i zapisanie pliku:

```
plik->Close();
```

Zadanie

Stworzyć plik `histogramy.root` zawierający:

- histogram zmiennej m,
- histogram zmiennej $x1$,
- histogram zmiennej m pod warunkiem, że $x1 > 1$,
- histogram zmiennej x pod warunkiem, że $x1 > 1$.

Podpowiedź: należy zmodyfikować makro `pierwszy_program_ROOT.C`

Zadanie dodatkowe:
Do pliku `histogramy.root`:

- dodać histogram dwuwymiarowy zmiennych $x1$ oraz $x2$,
- dodać histogram jednowymiarowy zmiennej $x1$ z wagą 1000.

Histogramy tworzone przez ROOTa nie są najpiękniejsze – czas to zmienić!

Otwarcie pliku i zadeklarowanie histogramu:

```
TFile * f = new TFile("histogramy.root","READ");
TH1F * h;
f->GetObject("nazwa_histogramu", h);
```

TCanvas – kanwa do rysowania

```
TCanvas *c = new TCanvas("nazwa","tytul");
```

Ustawienie rysowania w tej kanwie

```
c->cd();
```

Narysowanie histogramu na kanwie:

```
hh->Draw();
```

Narysowanie drugiego histogramu na tej samej kanwie:

```
h->Draw("same");
```

Zapisanie okna jako rysunku:

```
c->SaveAs("histogram.png");
```

Zapisanie okna jako rysunku w formacie wektorowym:

```
c->SaveAs("histogram.eps");
```

Do wyświetlania plików $*.eps$ służy np. program GhostView:

```
gv -watch nazwa_pliku.eps&
```

Upiększanie histogramów – cz. 1

Zmiana lewego marginesu kanwy:

```
c1->SetLeftMargin(0.14);
```

Zmiana osi y na logarytmiczną:

```
gPad->SetLogy(1);
```

Zmiana wielkości czcionki podpisu osi x:

```
h->GetXaxis()->SetTitleSize(0.05);
```

Zmiana odległości między osią x a podpisem:

```
h->GetXaxis()->SetTitleOffset(1.25);
```

Zmiana wielkości czcionki liczb na skali osi x:

```
h->GetXaxis()->SetLabelSize(0.05);
```

Zmiana odległości między osią x liczbami na skali:

```
h->GetXaxis()->SetLabelOffset(0.01);
```

Ustawienie maksimum na osi y:

```
h->SetMaximum(2.e-2);
```

Ustawienie tytułu wykresu:

```
h->SetTitle("opis");
```

Ustawienie tytułu osi x:

```
h->GetYaxis()->SetTitle("opis");
```

Zmiana koloru linii:

```
h->SetLineColor(2);
```

Zmiana stylu linii:

```
h->SetLineStyle(1);
```

Uwaga: powiązanie cyfr z kolorami/stylami można znaleźć w Internecie.

Zmiana grubości:

```
h->SetLineWidth(3);
```

Dodanie tekstu do obrazka:

```
TLatex tekst;
tekst.SetTextSize(0.05);
tekst.DrawLatex(pozx, pozy, "#sqrt{s} = 14 TeV");
```

pozx – początek napisu względem osi X
pozy – początek napisu względem osi Y

Legenda:

```
TLegend *legenda = new TLegend(0.75,0.8,0.97,0.97);
legenda->AddEntry(h, ópis wejscia", ł");
```

opcje: l – linia, p – punkt, f – prostokąt, e – błąd na osi Y

```
legenda->SetTextSize(0.05);
legenda->SetTextFont(42);
legenda->Draw("same");
```

Procedura *fitowania* pozwala na dopasowanie funkcji do danych. W programie ROOT można z niej korzystać na kilka sposobów:

1. Tryb interaktywny:
   - otworzyć plik `.root` zawierający histogramy,
   - uruchomić `TBrowser`,
   - wyświetlić histogram,
   - wybrać `Tools -> FitPanel`,
2. Przez polecenie, w przypadku gdy funkcja jest predefiniowana w programie ROOT:

```
h->Fit("gaus");
```

Znane funkcje:

- **gaus** – funkcja Gaussa z trzema parametrami (p0, p1, p2): $f(x) = p0 \cdot exp(-0.5 \cdot ((x - p1)/p2)^2))$,
- **expo** – eksponenta z dwoma parametrami: $f(x) = exp(p0 + p1 \cdot x)$,
- **polN** – wielomian N-tego stopnia: $f(x) = p0 + p1 \cdot x + p2 \cdot x^2 + \ldots$.

3. Przez dopasowanie funkcji zdefiniowanej przez użytkownika:

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

Zadania

1. Za pomocą `FitPanel` dopasować krzywą Gaussa do rozkładu m.
Jakie są parametry fitu? Co one znaczą? Jakie są błędy?
2. Zmienić zakres fitowania na 2.9 – 3.1.
Czy dofitowane wartości uległy zmianie?
3. Napisać makro, które:
   - wczytuje histogram rozkładu m z pliku `histogramy.root`,
   - rysuje go na kanwie,
   - zmienia kolor linii na niebieski a jej styl na linię przerywaną,
   - dodaje następujący opis osi X: "masa ukladu, m [GeV]",
   - dodaje następujący opis osi Y: "liczba przypadkow",
   - fituje krzywą Gausa do histogramu,
   - zapisuje go w pliku `masa.eps`.
4. Zadanie dodatkowe: wypróbować pozostałe opcje rysowania.

Kompilacja makra za pomocą g++

W makrze `pierwszy_program_ROOT.C` dodać funkcję główną:

```
1  int main(){
2    pierwszy_program_ROOT a;
3    a.Loop();
4
5    return 0;
6  }
```

Potrzebne są biblioteki ROOTa. Nie znajdują się one w domyślnych katalogach przeszukiwanych przez g++, dlatego trzeba go o tym poinformować:
`g++ -I ‘root-config --incdir‘ -o program program.C ‘root-config --libs‘`

Zadania:
1. Sprawdzić, co robią polecenia `root-config --incdir` oraz `root-config --libs` wpisane w terminalu.
2. Zmodyfikować makro `pierwszy_program_ROOT.C`, tak by je można było skompilować g++.1