Pracownia Specjalistyczna

Wprowadzenie do pakietu ROOT
Część 2

- pliki – odczyt i zapis
- histogramy
- wykresy
- inne obiekty graficzne: markery, linie, legendy…

dr Katarzyna Rusiecka
Zakład Fizyki Hadronów IF UJ
katarzyna.rusiecka@uj.edu.pl

Zadania z poprzednich zajęć

Zadanie 1:
W folderze roboczym utwórz plik o nazwie rootlogon.C. Zadaniem skryptu rootlogon ma być ładowanie następujących bibliotek:

- libHist.so
- libMathMore.so
- libMathCore.so
- libTree.so
- libPhysics.so
- libMatrix.so

W zależności od tego, czy dana biblioteka została załadowana poprawnie czy nie, wypisz na ekran odpowiedni komunikat.

Zadanie 2:

- Utwórz plik WidmoCs.C
- Plik powinien zawierać funkcję o tej samej nazwie, której argumentem będzie nazwa pliku tekstowego. Ustal wartość domyślną argumentu na “Cs-137.dat”
- Wewnątrz funkcji otwórz plik tekstowy będący jej argumentem. Sprawdź czy plik został poprawnie otwarty i wypisz odpowiedni komunikat
- Odczytaj zawartość pliku. Aby upewnić się, czy dane zostały poprawnie odczytane wypisz je na ekranie

Makro to będzie punktem wyjścia do naszych kolejnych zajęć i ćwiczeń z histogramami.

Standardowe wejście/wyjście do pliku – przypomnienie

Pliki nagłówkowe, które należy dołączyć na początku kodu, jeśli korzystamy z operacji wejścia lub wyjścia:

```
#include <iostream>
#include <ifstream>
#include <ofstrem>
#include <fstream>
using namespace std;
```

- `<iostream>` → Strumień wejścia/wyjścia na ekran
- `<ifstream>` → Strumień wejścia do pliku (odczyt danych)
- `<ofstrem>` → Strumień wyjścia do pliku (zapis danych)
- `<fstream>` → Strumień wejścia/wyjścia do pliku

Przykład wypisu na ekran:

```
cout << "some text" << endl;
```

Strumień wejścia z klawiatury:

```
Int_t number;
cin >> number;
```

Tworzenie strumienia do/z pliku:

```
fstrem myfile;
myfile.open("file.txt", mode);
```

tryb	opis
ios::in	Otwórz plik do odczytu
ios::out	Otwórz plik do zapisu
ios::binary	Otwórz mod w modzie binarnym
ios::app	Dodawaj do aktualnej zawartości pliku
ios::trunc	Usuń zawartość pliku i zastąp ją nowymi danymi

Przykłady:

```
fstrem myfile;
myfile.open("logfile.txt", ios::out | ios::app);

ifstream input("data.txt");
```

```
file.is_open();
file.eof();
file.close();
```

Sesja robocza w środowisku ROOT - przypomnienie

Aby uruchomić ROOT należy wpisać w terminalu:

```
$ root
```

Opcje:
-b – batch mode, bez grafiki
-n – zignoruj rootlogon.C i rootlogoff.C
-q – zakończ sesję po wykonaniu makra
~~-l – nie pokazuj ekranu powitalnego~~
dir – jeśli dir jest istniejącym katalogiem, wejdź do niego przed uruchomieniem ROOT'a
--web=off – najnowszych wersjach, otwieraj TBrowser, zamiast RBrowser

Przykłady:

```
$ root --web=off
```

→ Przy wywołaniu GUI wyświetli się tradycyjny TBrowser

```
$ root -b -q macro.C > logfile.txt
```

→ ROOT będzie działał w trybie “batch”, “macro.C” zostanie wykonane, a output zostanie przekierowany do pliku “logfile.txt”. Po wykonaniu makra sesja zostanie zamknięta.

```
$ root 'macro.C(argument)'
```

→ Makro “macro.C” zostanie załadowane i wykonane z podanymi argumentami.

Ładowanie makra:

```
root [0] .L macro.C
```

```
root [0] new TBrowser
```

Ładowanie i wykonywanie makra:

```
root [0] .x macro.C
```

```
root [0] .q
```

Pliki ROOT

Konstruktor pliku ROOT

```
TFile *file = new TFile("filename.root","OPTION");
```

Opcja	Opis
NEW or CREATE	Stwórz nowy plik i otwórz go w celu zapisu do pliku. Jeśli plik już istnieje, nie zostanie otwarty.
UPDATE	Otwórz istniejący plik w celu zapisu do pliku. Jeśli plik nie istnieje, zostanie utworzony.
READ	Otwórz istniejący plik do odczytu (opcja domyślna).
RECREATE	Stwórz nowy plik. Jeśli plik już istnieje zostanie nadpisany.

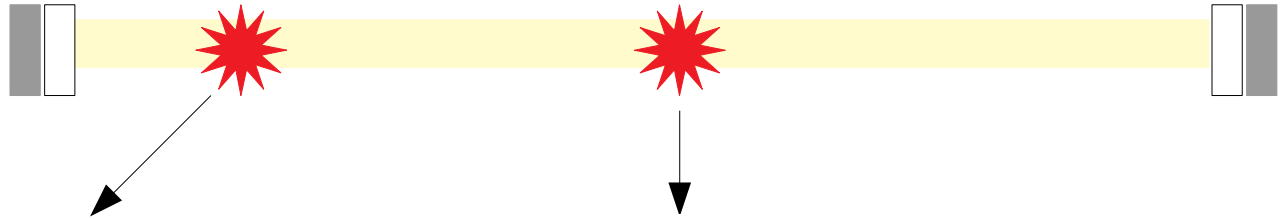
Przykład tworzenia katalogu i zapisywania histogramu w pliku ROOT:

```
TDirectory *dir = file->mkdir("directory");
dir->cd();
histogram->Write();
file->cd();
```

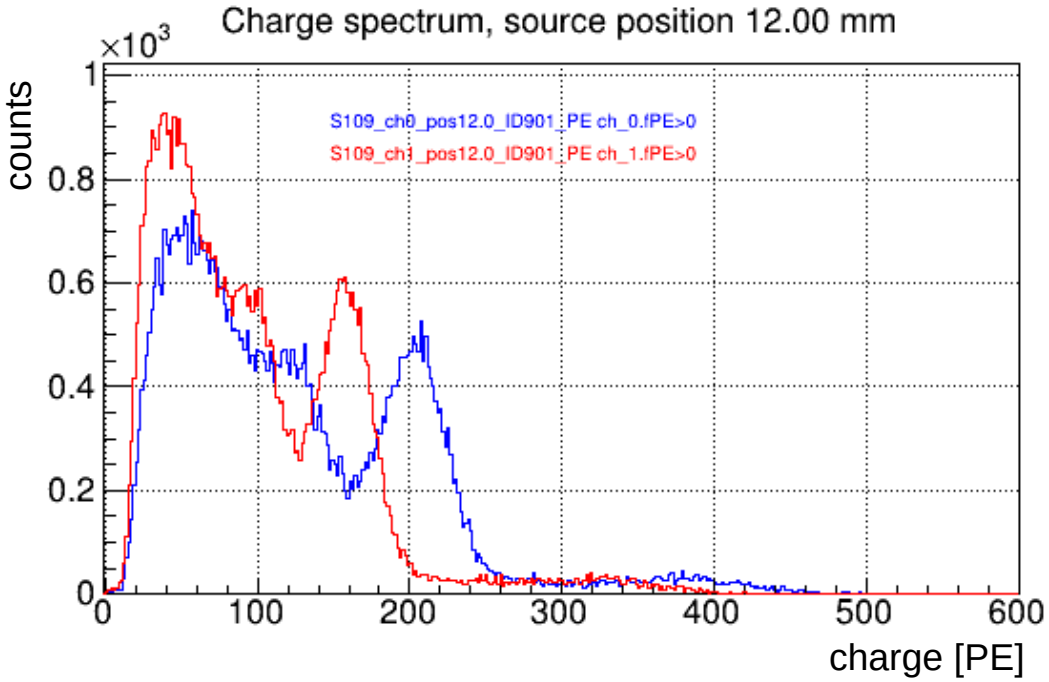
Histogramy

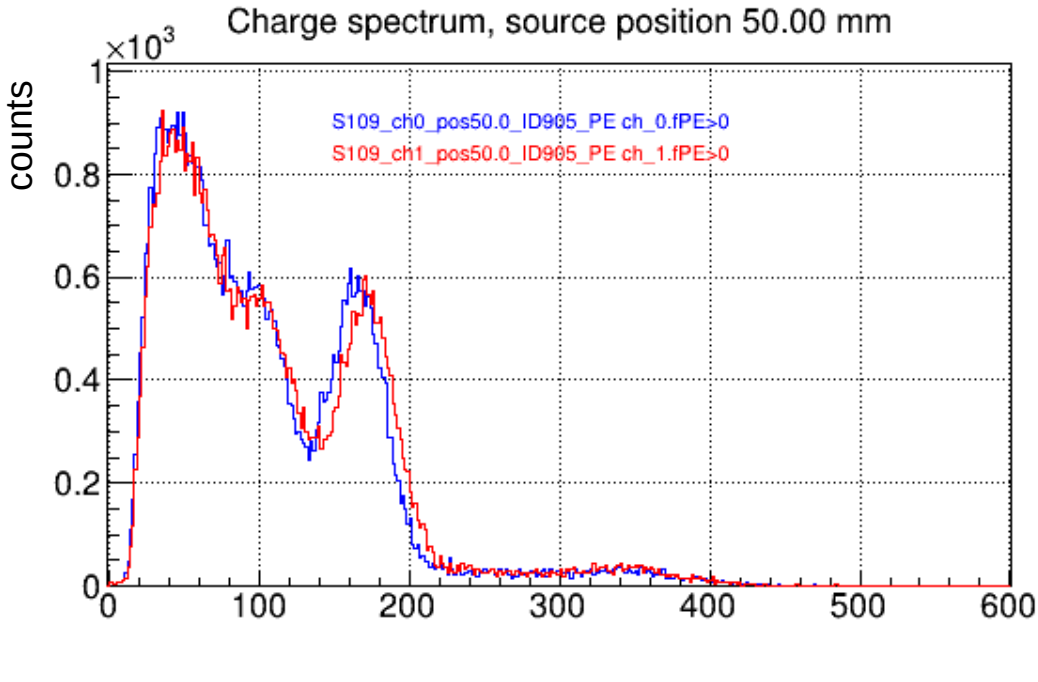
Przykłady histogramów wykonanych za pomocą pakietu ROOT:

Widma promieniowania gamma (Na-22) zarejestrowane za pomocą detektora scyntylacyjnego (włókno scyntylacyjne + fotopowielacze krzemowe). Widma zarejestrowano dla różnych pozycji źródła promieniotwórczego wzdłuż włókna.



LYSO:Ce





Tworzenie histogramów

Konstruktor histogramu TH1F

```
TH1F *hist = new TH1F(”name”,”title”,bins,xlow,xup);
```

name – nazwa histogramu, typ: char lub TString
title – tytuł histogramu, typ: char lub TString
bins – liczba binów w histogramie, typ: Int_t
xlow – dolna krawędź pierwszego binu
xup – górna krawędź ostatniego binu

Konstruktor histogramu TH2F

```
TH2F *hist = new TH2F(”name”,”title”,binsx,xlow,xup,binsy,ylow,yup);
```

binsx – liczba binów wzdłuż osi X
xlow – dolna krawędź pierwszego binu na osi X
xup – górna krawędź ostatniego binu na osi X
binsy – liczba binów wzdłuż osi Y
ylow – dolna krawędź pierwszego binu na osi Y
yup – górna krawędź ostatniego binu na osi Y

- Standard numerowania binów:
  - Bin = 0 → underflow bin
  - Bin = 1 → pierwszy bin z dolną krawędzią xlow
  - Bin = bins → ostatni bin z górną krawędzią xup
  - Bin = bins+1 → overflow bin

- Dla histogramów 2D i 3D – zdefiniowany jest globalny numer binu; np. dla histogramu 2D:

```
Int_t gbin = hist→GetBin(binx,biny);
```

Wypełnianie histogramów

Metoda 1 – Fill()

```
(…)

Int_t nbins = 100;

TH1F *hGaussDist = new TH1F(”hGaussDist”, ”hGaussDist”, nbins, -10, 10);

Int_t maxEvents = 5000;
Double_t random = 0;

for(Int_t i=0; i<maxEvents; i++){
  random = gRandom->Gaus(0, 2);
  hGaussDist->Fill(random);
}

(…)
```

Wypełnianie histogramów

Metoda 2 – SetBinContent()

```
(…)

const Int_t nbins = 1000;

TH1F *hSpectrum = new TH1F(”hSpectrum”, ”hSpectrum”, nbins, 0, nbins);

Double_t counts[nbins]; // tablica zawierające liczbę zliczeń w kanałach
Double_t errors[nbins]; // tablica zawierająca niepewności pomiarowe w kanałach

for(Int_t i=1; i<=1000; i++){
  hSpectrum->SetBinContent(i, counts[i]);
  hSpectrum->SetBinError(i, errors[i]);
}

(…)
```

Rysowanie

Canvas (TCanvas) to obszar (okienko), w którym rysowane są obiekty graficzne, takie jak histogramy, grafy, itp.

Tworzenie canvasu:

```
Int_t height = 500;
Int_t width = 500;
TCanvas *can = new TCanvas("name", "title", width, height);
```

- Jeśli canvas nie zostanie stworzony przed narysowaniem histogramu – zostanie on automatycznie stworzony

- W jednej sesji ROOT'a można otwierać wiele canvasów (pod warunkiem, że nazywają się inaczej)

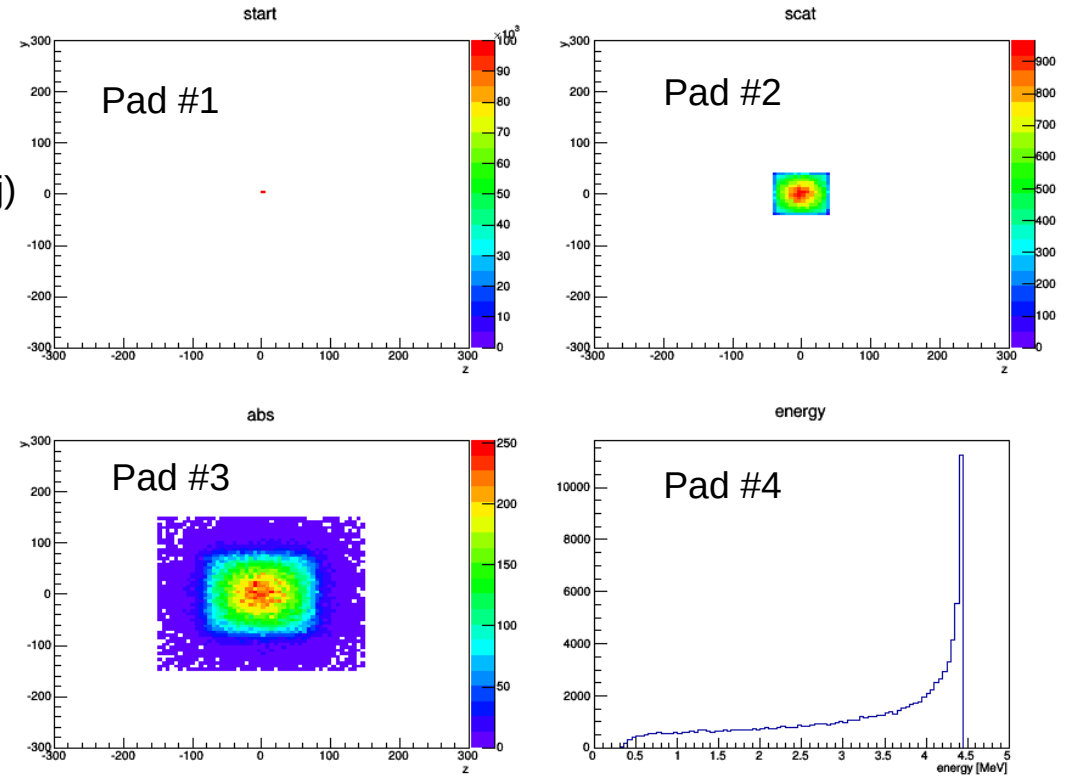
- Canvas można podzielić na mniejsze obszary - pady

Przykład podziału canvasu na pady:

```
can->Divide(2,2);
can->cd(1);
histogram->Draw();
```

- `Divide(liczba padów_w_rzędzie, liczba_padów_w_kolumnie)` lub `DivideSquare(liczba_padów)`

- Numerowanie padów zaczyna się od 1



Zadanie

Kontynuacja pracy nad makrem WidmoCs.C:

- Utwórz histogram oraz wypełnij go danymi, które odczytujesz z pliku Cs-137.dat
- Przypisz każdemu binowi odpowiednią niepewność pomiarową
- Sformatuj utworzony histogram:
  - Nazwij odpowiednio osie i histogram
  - Ustal odpowiednie zakresy osi
  - Ustal kolory, styl linii itp.
- Utwórz canvas i narysuj na nim histogram. Jak jeszcze możesz poprawić czytelność histogramu?
- Jak należałoby zmodyfikować kod, gdyby plik zawierający dane pomiarowe zapisano w formacie binarnym? Zapisz alternatywny fragment kodu jako komentarz w makrze (***)
- Utworzony canvas oraz histogram zapisz w pliku ROOT

Przykład

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

Bool_t DrawHistogram(void){

    TH1F *hGauss = new TH1F("gauss","gauss",50,-10,10);
    hGauss->FillRandom("gaus",5000);


    TFile *file = new TFile("histograms.root","UPDATE");

    if(!file->IsOpen()){
     cout << "Could not open output file!" << endl;
     return kFALSE;
    }

    hGauss->SetLineColor(kGreen+3);
    hGauss->SetLineWidth(2);
    hGauss->SetFillColor(kGreen-6);

    hGauss->GetXaxis()->SetTitle("x values");
    hGauss->GetYaxis()->SetTitle("y values");
    hGauss->SetTitle("Gaussian distribution");

    TCanvas *can = new TCanvas("can","can",600,600);
    can->cd();
    hGauss->Draw();

    hGauss->Write();
    file->Close();

    return kTRUE;
}
```

Załączanie plików nagłówkowych i deklaracja użycia przestrzeni nazw

Początek funkcji

Tworzenie histogramu i wypełnianie go zgodnie z rozkładem Gaussa

Tworzenie pliku ROOT

Sprawdzenie czy plik został poprawnie otworzony

Ustawianie atrybutów histogramu

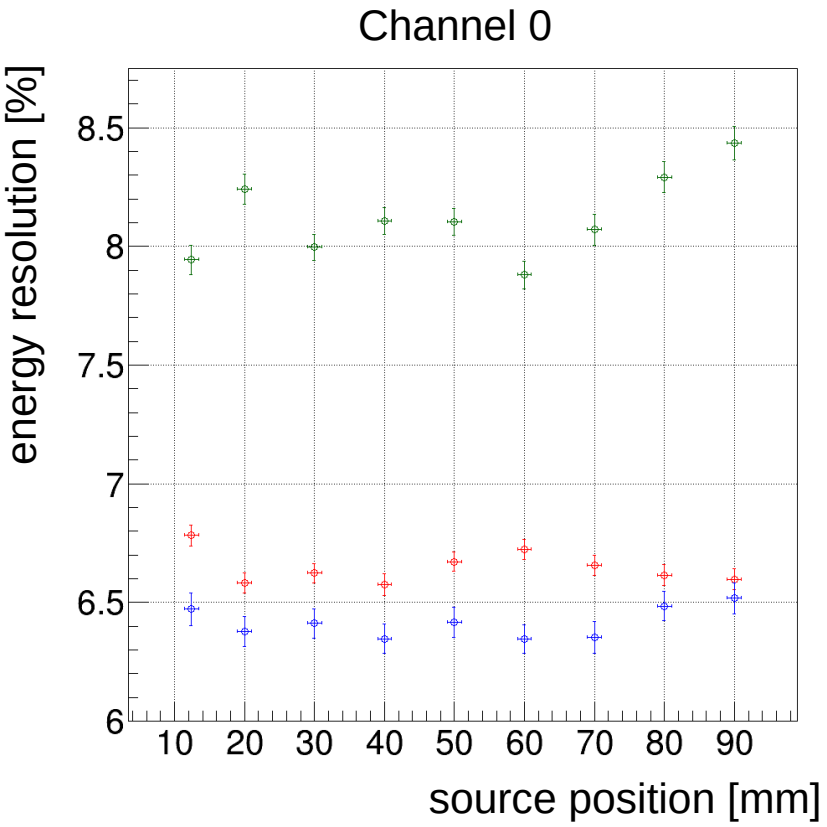
Tworzenie canvasu i rysowanie histogramu

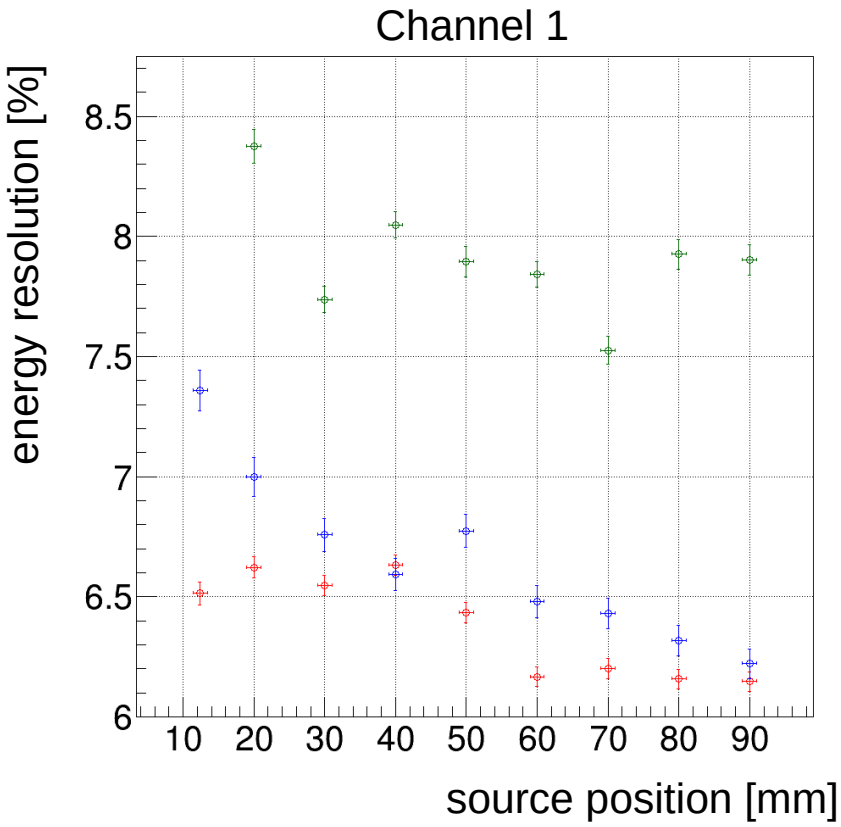
Zapisywanie do pliku i zamykanie pliku

Wykresy

Przykłady wykresów wykonanych za pomocą pakietu ROOT:

Energetczna zdolność rozdzielcza detektora scyntylacyjnego (włókno scyntylacyjne + fotopowielacz krzemowy) w funkcji położenia źródła wzdłuż detektora. Dane dla trzech materiałów scyntylacyjnych.





$ER_{ch0\ LYSO:Ce}$ = 6.65(1) % $ER_{ch1\ LYSO:Ce}$ = 6.37(1) %

$ER_{ch0\ GAGG:Ce}$ = 6.41(2) % $ER_{ch1\ GAGG:Ce}$ = 6.60(2) %

$ER_{ch0\ LuAG:Ce}$ = 8.10(2) % $ER_{ch1\ LuAG:Ce}$ = 8.07(2) %

Tworzenie wykresów – TGraph i TGraphErrors

TGraph konstruktor

```
TGraph *graph = new TGraph(npoints,x,y)
```

```
TGraph *graph = new TGraph(npoints)
```

`npoints` – liczba punktów na wykresie
x – tablica zawierająca wartości X
y – tablica zawierająca wartości Y

Jeśli konstruktor nie zawiera tablic z odpowiednimi wartościami, wykres należy wypełnić przy pomocy metody `SetPoint(i,x,y)` - `i` – numer punktu, x – wartość X, y – wartość Y.

TGraphErrors konstruktor

```
TGraphErrors *graph = new TGraphErrors(npoints,x,y,ex,ey)
```

`npoints` – liczba punktów na wykresie
x – tablica zawierająca wartości X
y – tablica zawierająca wartości Y
`ex` – tablica zawierająca niepewności X
`ey` – tablica zawierająca niepewności Y

- `TGraphErrors` może zostać stworzony także przy pomocy “krótkiego” konstruktora, tzn. podajemy tylko liczbę punktów na wykresie. Wykres należy wtedy wypełnić za pomocą metod `SetPoint(i,x,y)` oraz `SetPointError(i,ex,ey)`.
- Jeśli nie chcemy niepewności na wykresie, zamiast `ex` i `ey` w konstruktorze wpisujemy 0.

Tworzenie wykresów – TGraph, TGraphErrors oraz wykresy 2D

Konstruktor “z pliku”

```
TGraph *graph = new TGraph(“filename.txt”,”format”,”opcja”)
```

`filename.txt` – nazwa pliku z danymi
`format` – sposób odczytywania danych z pliku, np. `“%lg %lg”` oznacza odczytywanie dwóch pierwszych kolumn liczb w pliku i przypisywanie ich odpowiednio wartościom X i Y. Jeśli chcemy pominąć kolumnę - `“%*lg”`
`opcja` – dotyczy sytuacji, kiedy kolumny liczb są rozdzielone np. znakami interpunkcyjnymi. Jeśli kolumny są rozdzielone za pomocą średników, wpisujemy `“;”`.

Wykresy 2D – konstruktory

```
TGraph2D *graph = new TGraph2D(npoints,x,y,z)
```

```
TGraph2DErrors *graph = new TGraph2DErrors(npoints,x,y,z,ex,ey,ez)
```

`npoints` – liczba punktów na wykresie
`x` – tablica zawierająca wartości X
`y` – tablica zawierająca wartości Y
`z` – tablica zawierająca wartości Z
`ex` – tablica zawierająca niepewności X
`ey` – tablica zawierająca niepewności Y
`ez` – tablica zawierająca niepewności Z

• Dla wykresów 2D istnieje również konstruktor “z pliku”. Używając tego konstruktora należy pamiętać o odpowiednim formacie.

• Rysowanie wykresów – klasa `TGraphPainter`

Zadanie

Napisz makro Wykresy.C , w którym:

- Odczytasz dane z rozkładami kątowymi wygenerowane przez program TALYS (pliki `pp*ang.L1`)
- Dane z każdego pliku narysujesz w formie osobnego wykresu TGraph
- Wszystkie wykresy narysujesz na jednym canvasie i tym samym padzie
- Pamiętaj o odpowiednim formatowaniu wykresów:
  - ustaleniu odpowiednich zakresów osi
  - nadaniu tytułów osiom i wykresom
  - przypisaniu wykresom unikalnych markerów i kolorów, aby były rozróżnialne
  - jak jeszcze możesz poprawić czytelność wykresu?
- Utworzony canvas oraz wszystkie wykresy zapisz w pliku ROOT

Komentarz:

- Wygenerowane rozkłady kątowe dotyczą reakcji $^{12}C(p,p'\ gamma_{4.44})^{12}C$
- Liczba zawarta w nazwie wygenerowanych plików oznacza energię protonów bombardujących jądra węgla
- Przy rysowaniu wykresów będziemy potrzebować pierwszych dwóch kolumn liczb: angle (kąt a stopniach) oraz xs (przekrój czynny w jednostkach arbitralnych)

Przykład

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

Bool_t Wykres(Double_t energia){

  const int n = 91;
  TString nazwa = Form("pp0%.3fang.L01",energia);

  fstream dane(nazwa, ios::in);

  if(!dane.is_open()){
    cout << "Nie mozna otworzyc pliku tekstowego" <<  endl;
    return kFLASE;
  }

  Double_t katy[n];
  Double_t pczynny[n];
  Double_t dummy;
  TString naglowek;

  for(Int_t i=0; i<5; i++){
   naglowek.ReadLine(dane);
  }

  for(Int_t j=0; j<n; j++){
    dane >> katy[j] >> pczynny[j] >> dummy >> dummy;
  }
```

Początek funkcji

Konstruowanie nazwy pliku tekstowego

Otwieranie pliku tekstowego do odczytu

Sprawdzanie czy plik został poprawnie otwarty

Tablice i zmienne pomocnicze do przechowywania danych

Czytanie pliku

cdn...

Przykład cd

```
  TGraph *graph = new TGraph(n,katy,pczynny);

  graph->SetMarkerStyle(8);
  graph->SetMarkerColor(kRed);
  graph->SetMarkerSize(0.5);
  graph->SetLineColor(kGreen);
  graph->SetLineWidth(1);

  graph->Draw("APC");

  return kTRUE;
}
```

Tworzenie wykresu

Ustalanie atrybutów wykresu

Rysowanie wykresu

Legenda

Legenda – konstruktor:

```
TLegend *leg = new TLegend(x1,y1,x2,y2,"header","option");
```

`x1,y1,x2,y2` – współrzędne legendy (znormalizowane) na aktywnym padzie
`header` – tytuł legendy
`option` – opcje np. granice, cień, tło, itp.

Dodawanie obiektów do legendy:

```
leg->AddEntry(pointer,"text","option");
```

`pointer` – wskaźnik do wykresu, histogramu lub funkcji
`text` – opis
`option` – P – punkt, L – linia, E - niepewność

Rysowanie legendy

```
leg->Draw();
```

• Legendę można wygenerować automatycznie klikając prawym przyciskiem myszy na pad i wybierając opcję `BuildLegend`. Taka legenda będzie zawierała wszystkie obiekty narysowane na danym padzie opisane za pomocą ich tytułów.

Obiekty graficzne: markery, linie, tekst

Przykład rysowania linii

```
TLine line;
line.SetLineColor(kGreen);
line.SetLineWidth(1);
line.DrawLine(40,0,40,25);
line.DrawLine(160,0,160,25);
```

Właściwości graficzne linii są opisane przez klasę TAttLine

Przykład rysowania markera

```
TMarker marker;
marker.SetMarkerStyle(2);
marker.SetMarkerSize(2);
marker.SetMarkerColor(kBlue);
marker.DrawMarker(10,25);
```

Właściwości graficzne markerów są opisane przez klasę TAttMarker

Przykład umieszczania tekstu na padzie

```
TLatex text;
text.SetTextSize(0.02);
text.SetTextColor(kBlack);
text.SetTextFont(42);
text.DrawLatex(100,10,"#splitline{this
graph shows comparison}{of 2 angular
distributions}");
```

Właściwości graficzne tekstu są opisane przez klasę TAttText

Zadanie

Część 1:
Zmodyfikuj makro WidmoCs.C dodając do histogramu obiekty graficzne:

- Narysuj linie, które będą wskazywały na położenie ważnych struktur na widmie, np. Fotopik, krawędź Comptonowska
- Dodaj opisy zaznaczonych struktur
- Zapisz zmodyfikowane obiekty w pliku ROOT

Część 2:
Zmodyfikuj makro Wykresy.C w następujący sposób:

- Dodaj legendę, która pozwoli na rozróżnienie wykresów i poprawi czytelnośc obrazka
- Zapisz zmodyfikowane obiekty w pliku ROOT