From the TTask1.h file: declaration of the variables

```
#include <vector>
#include <map>

#include "TH1D.h"
```

std::vector documentation: http://www.cplusplus.com/reference/vector/vector/vector/

https://en.cppreference.com/w/cpp/container/vector

std::map documentation: http://www.cplusplus.com/reference/map/map/

https://en.cppreference.com/w/cpp/container/map

```
ULong64_t mEventId;

size_t mNbOfEvents;

std::map<ULong64_t, std::vector<Double_t>> mPrimaryVertices;
std::map<ULong64_t, std::vector<Double_t>> mSecondaryVertices;

std::map<ULong64_t, std::vector<std::vector<Double_t>>> mDaughterTrackLineSets;

TH1D* mH1_CharmedParticleFlightLengths;
TH1D* mH1_DaughterTrackIPs;
```

From the TTask1.cpp file: initialization of the histograms

```
void TTask1::InitHists()
{
  TH1D* h1;

  h1 = mH1_CharmedParticleFlightLengths = new TH1D("H1_CharmedParticleFlightLengths",
                                                   "Flight lengths of charmed hadrons",
                                                   9, 0, 5300);

  h1->SetFillColor(kBlue); // h1 points to mH1_CharmedParticleFlightLengths now,
                           // So, this is equivalent to mH1_CharmedParticleFlightLengths->SetFillColor(kBlue);

  h1->SetXTitle("Flight length (#mum)"); // #mu stands for a greek letter
  h1->SetYTitle("Nb of events");

  //---

  h1 = mH1_DaughterTrackIPs = new TH1D("H1_DaughterTrackIPs",
                                       "Impact parameters of tracks of the daughter particles",
                                       10, 0, 500);

  h1->SetFillColor(kRed);  // h1 points to mH1_DaughterTrackIPs now,
                           // So, this is equivalent to mH1_DaughterTrackIPs->SetFillColor(kRed);

  h1->SetXTitle("IP (#mum)");
  h1->SetYTitle("Nb of tracks");

  //---
}
```

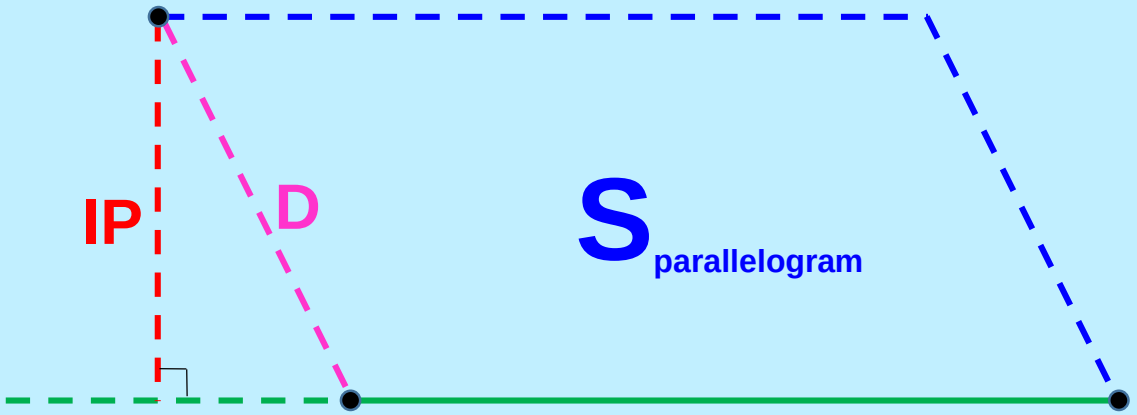
From the TTask1.cpp file: reading the *_TrackLines.csv files

```
void TTask1::ReadTrackLinesFile(const string& DataFilePath)
{
  ifstream ifs(DataFilePath);

  if (!ifs)
  {
    cerr << "Error in TTask1::" << __func__ << "():" << endl
         << "Can't open file " << DataFilePath << "!" << endl
         << "Exit..." << endl;
    exit(0);
  }

  string fstring;

  getline(ifs, fstring);

  vector<vector<Double_t>>& DTLS = mDaughterTrackLineSets[mEventId];

  while ( getline(ifs, fstring) )
  {
    istringstream istr1(fstring);

    size_t trType;

    istr1 >> trType;

    if (trType != 10) continue; // skips all tracks except for the daughter particle ones!

    istr1.ignore(); // skips the comma

    vector<Double_t> DTLC(6);

    for (Int_t i = 0; i < 6; i++)
    {
      istr1 >> DTLC[i];
      istr1.ignore();
    }

    DTLS.emplace_back(DTLC);
  }

  ifs.close();
}
```

Distances in 3D space

Primary ν int. vertex:

$V_0\ (x_0, y_0, z_0)$



Secondary interaction vertex and also the first point of the daughter particle track:

$V_1\ (x_1, y_1, z_1)$

Second point of the daughter particle track:

$P_2\ (x_2, y_2, z_2)$

In these notations, for example, vector $\overline{V_0V_1}$ looks like this:

$$\overline{V_0V_1} = (x_1 - x_0, y_1 - y_0, z_1 - z_0) = (dx_{10}, dy_{10}, dz_{10})$$

And the formulas of interest to us are as follows.

a) Calculation of the decay length:

$$D \equiv |\overline{V_0V_1}| = \sqrt{dx_{10}^2 + dy_{10}^2 + dz_{10}^2}$$

b) Calculation of the impact parameter:

$$S_{parallelogram} = |\overline{V_0V_1} \times \overline{V_1P_2}| = IP * |\overline{V_1P_2}|$$

$$IP = \frac{|\overline{V_0V_1} \times \overline{V_1P_2}|}{|\overline{V_1P_2}|} = \frac{\begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ dx_{10} & dy_{10} & dz_{10} \\ dx_{21} & dy_{21} & dz_{21} \end{vmatrix}}{\sqrt{dx_{21}^2 + dy_{21}^2 + dz_{21}^2}} =$$

$$= \sqrt{\frac{(dy_{10}dz_{21} - dy_{21}dz_{10})^2 + (dx_{10}dz_{21} - dx_{21}dz_{10})^2 + (dx_{10}dy_{21} - dx_{21}dy_{10})^2}{dx_{21}^2 + dy_{21}^2 + dz_{21}^2}}$$

In order to make formulas for calculation of the decay length (D) and the impact parameter (IP) more compact, let's introduce the following notations:

$$dx_{ij} \equiv x_i - x_j,$$
$$dy_{ij} \equiv y_i - y_j,$$
$$dz_{ij} \equiv z_i - z_j.$$
$$(i, j = 0, 1, 2)$$

From the TTask1.cpp file: filling the histograms

```
#include "Riostream.h"
#include "TFile.h"
#include "TSystemDirectory.h"
#include "TSystem.h"

#include "TMath.h"

#include "TTask1.h"
```

TMath namespace reference:

root.cern.ch/doc/master/namespaceTMath.html

```
void TTask1::FillHist_CharmedParticleFlightLengths()
{
  for (const auto& PrimaryVertices : mPrimaryVertices)
  {
    const ULong64_t& EventId = PrimaryVertices.first;

    const vector<Double_t>& V0 = PrimaryVertices.second;
    const vector<Double_t>& V1 = mSecondaryVertices[EventId];

    Double_t dx10 = V1[0] - V0[0];
    Double_t dy10 = V1[1] - V0[1];
    Double_t dz10 = V1[2] - V0[2];

    Double_t D = TMath::Sqrt(dx10*dx10 + dy10*dy10 + dz10*dz10);

    mH1_CharmedParticleFlightLengths->Fill(D);
  }
}
//---------------------------------------------------------------------------

void TTask1::FillHist_DaughterTrackIPs()
{
  for (const auto& PrimaryVertices : mPrimaryVertices)
  {
    const ULong64_t& EventId = PrimaryVertices.first;

    const vector<Double_t>& V0 = PrimaryVertices.second;

    const vector<vector<Double_t>>& DTLS = mDaughterTrackLineSets[EventId];

    for (const vector<Double_t>& DTLC : DTLS)
    {
      Double_t dx21 = DTLC[3] - DTLC[0]; // x2 - x1
      Double_t dy21 = DTLC[4] - DTLC[1]; // y2 - y1
      Double_t dz21 = DTLC[5] - DTLC[2]; // z2 - z1

      Double_t dx10 = DTLC[0] - V0[0];   // x1 - x0
      Double_t dy10 = DTLC[1] - V0[1];   // y1 - y0
      Double_t dz10 = DTLC[2] - V0[2];   // z1 - z0

      Double_t s1 = (dy21*dz10 - dy10*dz21);
      Double_t s2 = (dx21*dz10 - dx10*dz21);
      Double_t s3 = (dx21*dy10 - dx10*dy21);

      Double_t IP = TMath::Sqrt( (s1*s1 + s2*s2 + s3*s3)/(dx21*dx21 + dy21*dy21 + dz21*dz21) );

      mH1_DaughterTrackIPs->Fill(IP);
    }
  }
}
```

$$= \sqrt{dx_{10}^2 + dy_{10}^2 + dz_{10}^2}$$

$$= \sqrt{\frac{(dy_{10}dz_{21} - dy_{21}dz_{10})^2 + (dx_{10}dz_{21} - dx_{21}dz_{10})^2 + (dx_{10}dy_{21} - dx_{21}dy_{10})^2}{dx_{21}^2 + dy_{21}^2 + dz_{21}^2}}$$

From the DrawHist.C file: drawing the histograms

```
void DrawHists()
{
  gStyle->SetOptStat(111110);

  // Load histograms from a file

  TFile* HistFile = TFile::Open("HistFile.root");

  TH1D* H1_CharmedParticleFlightLengths = (TH1D*)HistFile->Get("H1_CharmedParticleFlightLengths");

  TH1D* H1_DaughterTrackIPs = (TH1D*)HistFile->Get("H1_DaughterTrackIPs");

  // Draw the histograms

  TCanvas* C1_CharmSampleHists = new TCanvas("C1_CharmSampleHists",
                                             "C1_CharmSampleHists",
                                             50, 50,     // Position of the upper-left corner (in pixels)
                                             1400, 700); // Width and height (in pixels)

  C1_CharmSampleHists->Divide(2, 1); // Divides the canvas into cells organized in one row and too columns

  C1_CharmSampleHists->cd(1); // Makes the first cell of the canvas active (to draw a histogram here)

  H1_CharmedParticleFlightLengths->Draw("hist");

  C1_CharmSampleHists->cd(2); // Makes the second cell of the canvas active (to draw a histogram here)

  H1_DaughterTrackIPs->Draw("hist");

  C1_CharmSampleHists->SaveAs("CharmSampleHists.png");
}
```