ROOT BASICS and ADVANCED

ABHISEK PRAHARAJ

WHAT IS ROOT?

It is and an open-source data analysis framework used by high energy physics and others.

ROOT is an object-oriented computer program and library developed by CERN.

It was originally designed for particle physics data analysis and contains several features specific to the field, but it is also used in other applications such as astronomy and data mining.

Checkpoints

- hISTOGRAM
- 2d hISTOGRAM
- gRAPH
- gRAPH pLOT wITH eRROR bARS
- cREATING uSER dEFINED fUNCTIONS
- sTORING aND lOADING oBJECT IN TFile
- TTree
- sTORING aND rEADING dATA fROM TTree

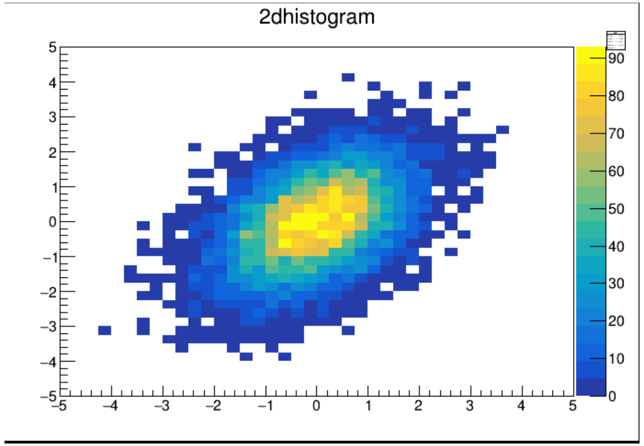
hISTOGRAM

A histogram is a bar graph-like representation of data that buckets a range of classes into columns along the horizontal x-axis.

The vertical y-axis represents the number count or percentage of occurrences in the data for each column.

1D histogram is nothing more than counting how many voxels with a particular intensity occur in the dataset.

While in case of the 2D histogram, The 2D histogram is mostly used to compare 2 channels in a multi-channel dataset, where the x-axis represent the intensities of the first channel and the y-axis the intensities of the second channel.



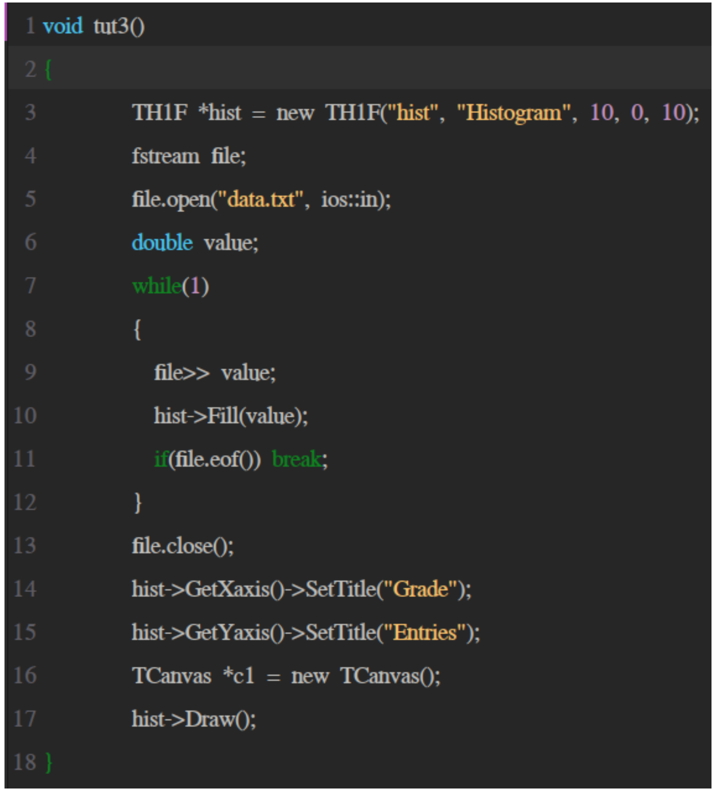
Histogram

1.Create a data file and Read the input file to a histogram.

Steps-

- Void macroname();
- {
- Create an histogram with bin size number and bin limit.
- Define its axis and give the title if needed.
- Optional- create a canvas and call it to draw.
- Open the created data file and and fill the histogram.
- Always close the file.
- }

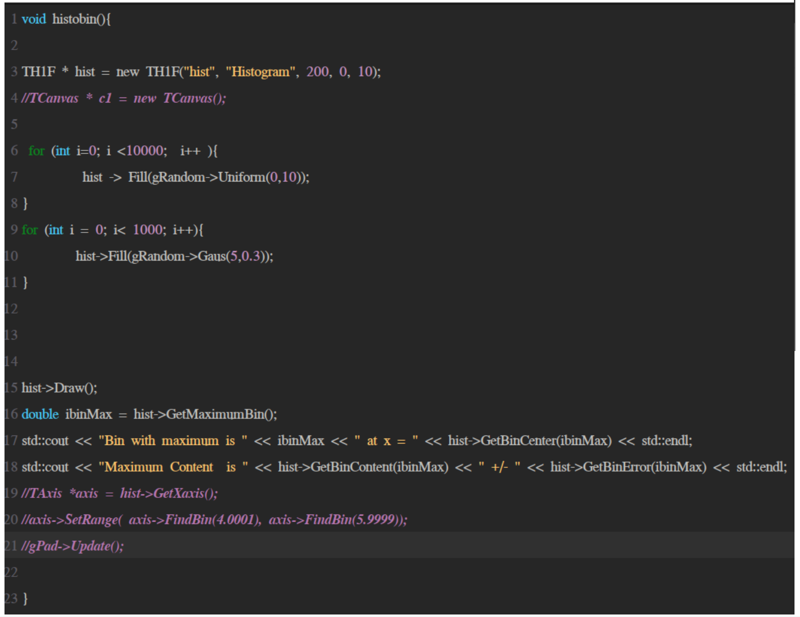
Code snippet



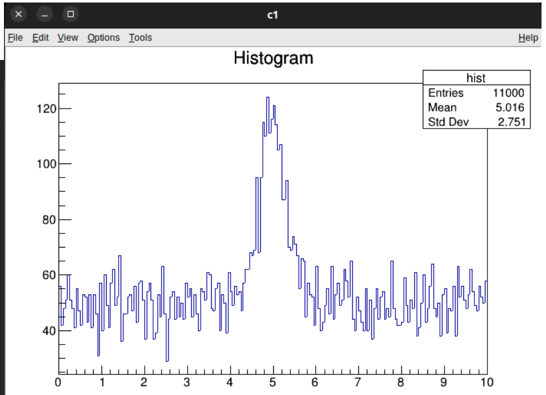
```
void tut3()
{
        TH1F *hist = new TH1F("hist", "Histogram", 10, 0, 10);
        fstream file;
        file.open("data.txt", ios::in);
        double value;
        while(1)
        {
          file>> value;
          hist->Fill(value);
          if(file.eof()) break;
        }
        file.close();
        hist->GetXaxis()->SetTitle("Grade");
        hist->GetYaxis()->SetTitle("Entries");
        TCanvas *c1 = new TCanvas();
        hist->Draw();
}
```

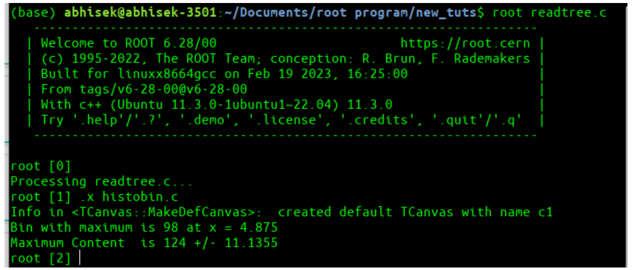
Histogram Exercises

1.Create an histogram with 200 bins between 0 and 10. Fill then the histogram with 1000 gaussian random numbers with mean 5 and sigma 0.3 and 10000 uniform number between 0 and 10. Plot this histogram. Find the bin number of the histogram with the maximum content. What is its bin content ? What is the bin center and the bin error ?



```
void histobin(){

TH1F * hist = new TH1F("hist", "Histogram", 200, 0, 10);
//TCanvas * c1 = new TCanvas();

 for (int i=0; i <10000;   i++ ){
            hist -> Fill(gRandom->Uniform(0,10));
}
for (int i = 0; i< 1000; i++){
          hist->Fill(gRandom->Gaus(5,0.3));
}



hist->Draw();
double ibinMax = hist->GetMaximumBin();
std::cout << "Bin with maximum is " << ibinMax << " at x = " << hist->GetBinCenter(ibinMax) << std::endl;
std::cout << "Maximum Content  is " << hist->GetBinContent(ibinMax) << " +/- " << hist->GetBinError(ibinMax) << std::endl;
//TAxis *axis = hist->GetXaxis();
//axis->SetRange( axis->FindBin(4.0001), axis->FindBin(5.9999));
//gPad->Update();

}
```

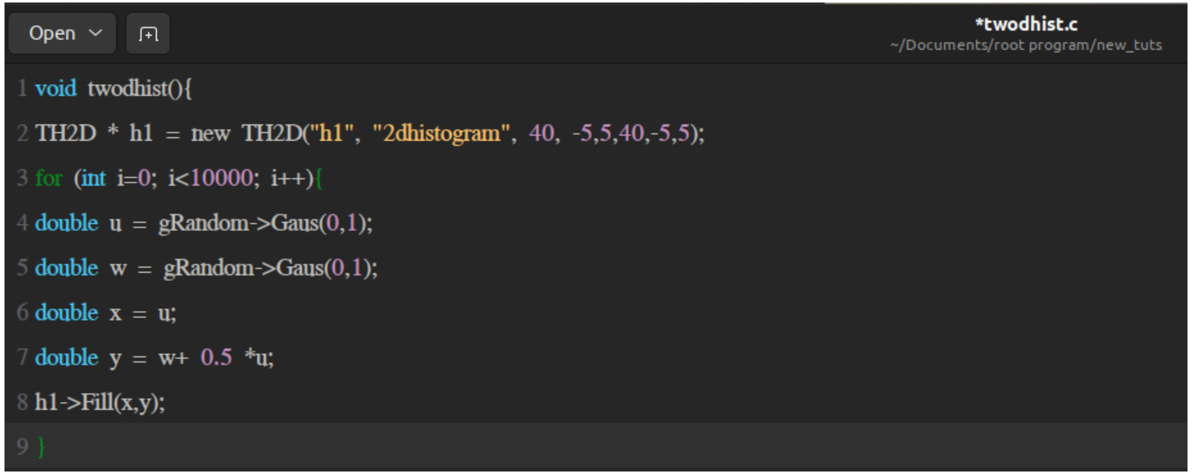




```
(base) abhisek@abhisek-3501:~/Documents/root program/new_tuts$ root readtree.c
   ------------------------------------------------------------------
  | Welcome to ROOT 6.28/00                        https://root.cern |
  | (c) 1995-2022, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Feb 19 2023, 16:25:00                 |
  | From tags/v6-28-00@v6-28-00                                      |
  | With c++ (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0                   |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------

root [0]
Processing readtree.c...
root [1] .x histobin.c
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
Bin with maximum is 98 at x = 4.875
Maximum Content  is 124 +/- 11.1355
root [2] 
```

2D Histogram

2.Create a 2 dimensional histogram with x and y in the range [-5,5] and [-5,5] and 40 bins in each axis. Fill the histogram with correlated random normal numbers. To do this generate 2 random normal numbers (mean=0, sigma=1) u and w. Then use x = u and y=w+0.5*u for filling the histogram.



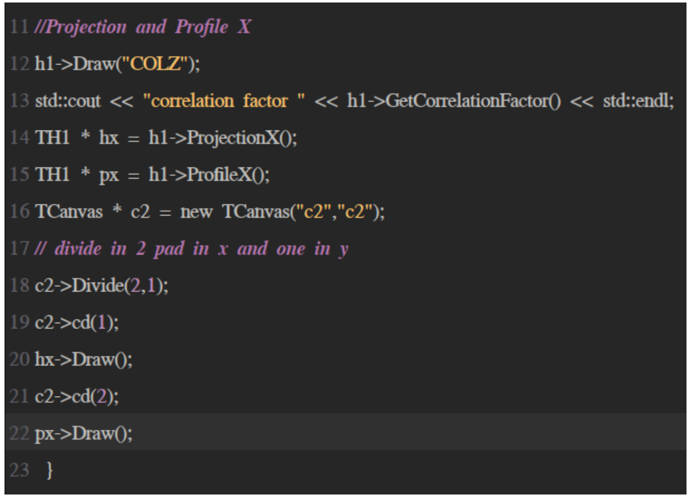
```
void twodhist(){
TH2D * h1 = new TH2D("h1", "2dhistogram", 40, -5,5,40,-5,5);
for (int i=0; i<10000; i++){
double u = gRandom->Gaus(0,1);
double w = gRandom->Gaus(0,1);
double x = u;
double y = w+ 0.5 *u;
h1->Fill(x,y);
}
```

Projection and Profile of 2D Histogram.

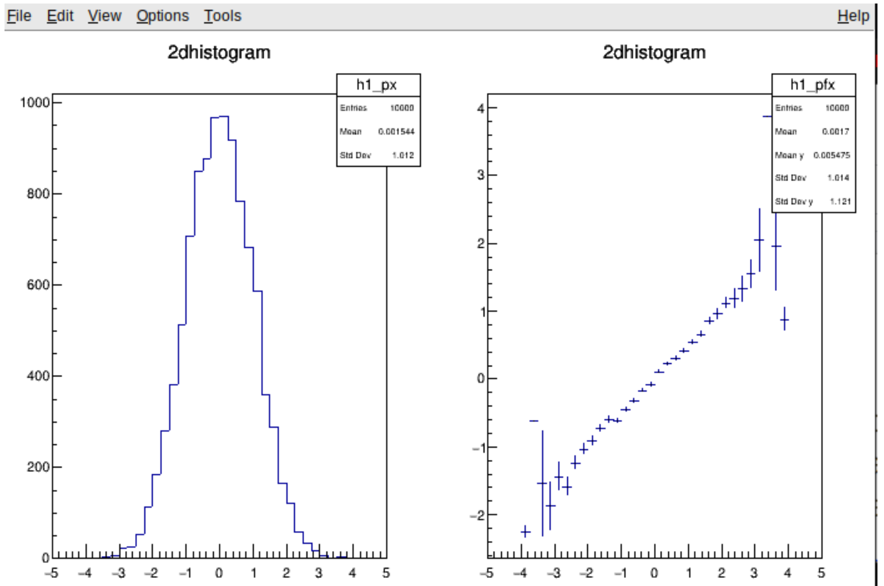
- In a 2D histogram, the projection refers to the distribution of values in one of the dimensions of the histogram. The projection is obtained by summing the values in the other dimension of the histogram.
- For example, suppose we have a 2D histogram of pixel intensity values in an image, with one dimension representing brightness values from 0 to 255, and the other dimension representing the frequency of each brightness value. To obtain the projection of the histogram along the brightness dimension, we would sum the frequency values for each brightness value. The resulting projection would show how frequently each brightness value occurs in the image.
- The profile of a 2D histogram refers to a cross-section of the histogram along a particular line or axis. This cross-section shows the distribution of values along the line or axis. The profile is obtained by summing the values along the line or axis in the 2D histogram.
- For example, if we have a 2D histogram of the intensity values of an image, the profile along a horizontal line through the middle of the image would show how the intensity values vary along that line. The profile along a vertical line through the middle of the image would show how the intensity values vary along that axis.

Code snippets and image

Correlation factor
0.453188



```
//Projection and Profile X
h1->Draw("COLZ");
std::cout << "correlation factor " << h1->GetCorrelationFactor() << std::endl;
TH1 * hx = h1->ProjectionX();
TH1 * px = h1->ProfileX();
TCanvas * c2 = new TCanvas("c2","c2");
// divide in 2 pad in x and one in y
c2->Divide(2,1);
c2->cd(1);
hx->Draw();
c2->cd(2);
px->Draw();
 }
```



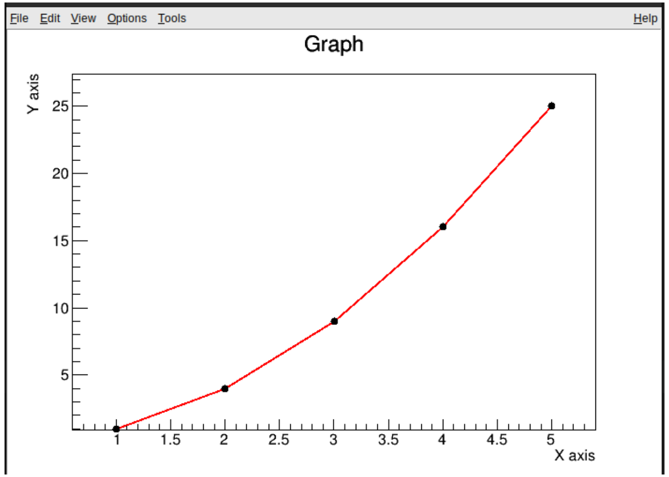
Correlation factor

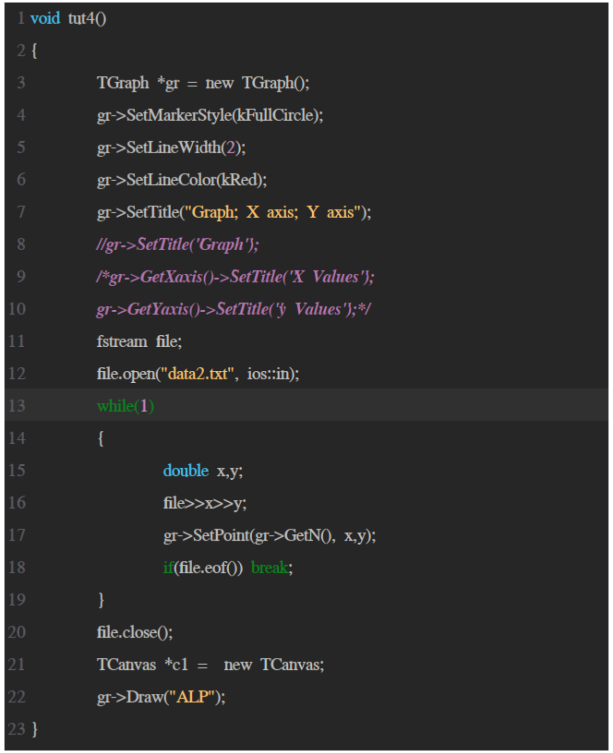
- Correlation factor, also known as correlation coefficient, is a statistical measure that describes the strength and direction of the linear relationship between two variables. It is represented by the symbol "r" and can range from -1 to +1, with values closer to -1 or +1 indicating a stronger correlation.
- A positive correlation means that as one variable increases, the other variable also tends to increase. A negative correlation means that as one variable increases, the other variable tends to decrease. A correlation of 0 indicates no linear relationship between the two variables.

Graph

4.Creating a data file and read that data file to a graph.

- Void macroname()
- {
- Define a Graph.
- Open the data file and using a loop read this data
- Close the file
- Call the canvas to draw the graph



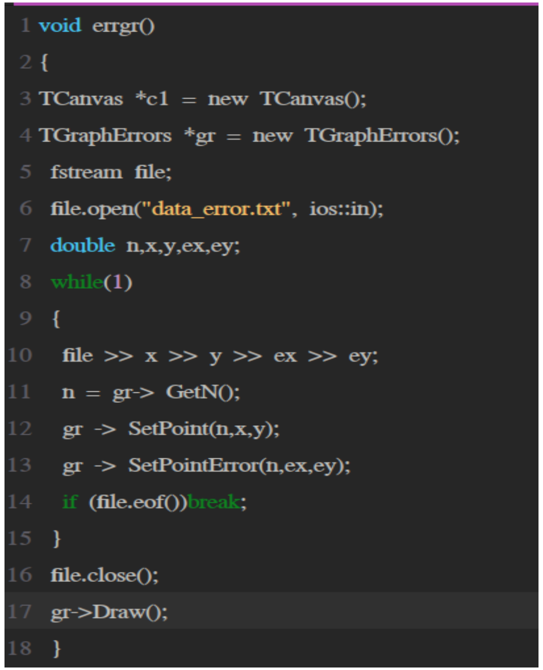


```
void tut4()
{
        TGraph *gr = new TGraph();
        gr->SetMarkerStyle(kFullCircle);
        gr->SetLineWidth(2);
        gr->SetLineColor(kRed);
        gr->SetTitle("Graph; X axis; Y axis");
        //gr->SetTitle('Graph');
        /*gr->GetXaxis()->SetTitle('X Values');
        gr->GetYaxis()->SetTitle('y Values');*/
        fstream file;
        file.open("data2.txt", ios::in);
        while(1)
        {
                double x,y;
                file>>x>>y;
                gr->SetPoint(gr->GetN(), x,y);
                if(file.eof()) break;
        }
        file.close();
        TCanvas *c1 = new TCanvas;
        gr->Draw("ALP");
}
```

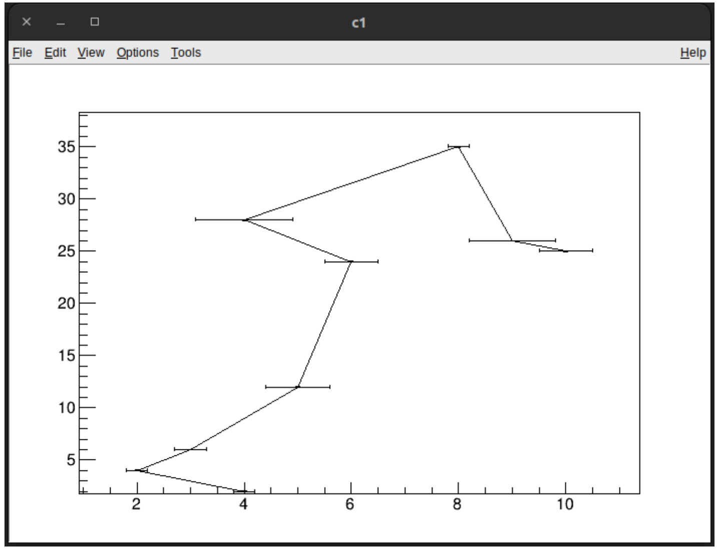
Plotting Data with Error Bars

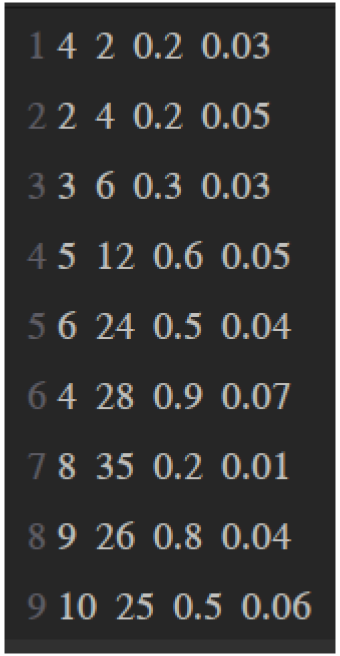
5.Create a data file with error values in data and plot it in graph.

Same as graph but we have to add 2 more extra variables for error values.



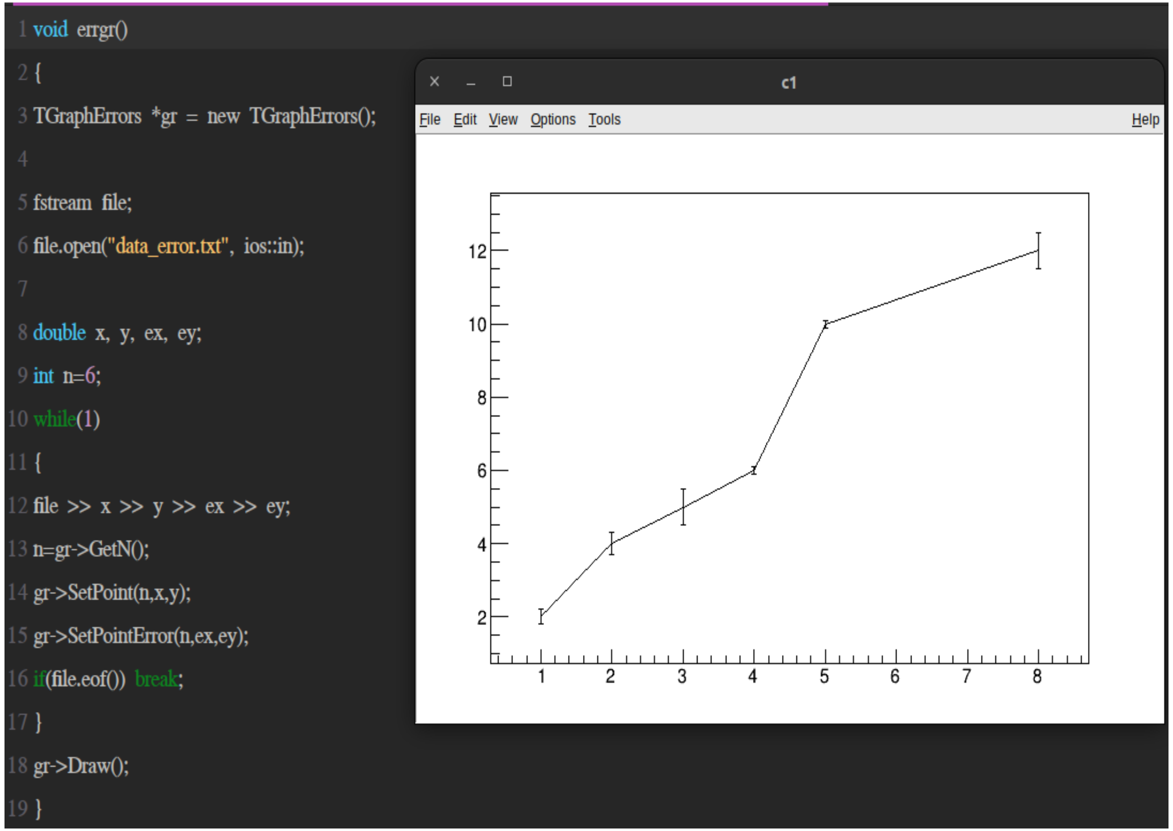
```
void errgr()
{
TCanvas *c1 = new TCanvas();
TGraphErrors *gr = new TGraphErrors();
 fstream file;
 file.open("data_error.txt", ios::in);
 double n,x,y,ex,ey;
 while(1)
 {
  file >> x >> y >> ex >> ey;
  n = gr-> GetN();
  gr -> SetPoint(n,x,y);
  gr -> SetPointError(n,ex,ey);
  if (file.eof())break;
 }
 file.close();
 gr->Draw();
 }
```



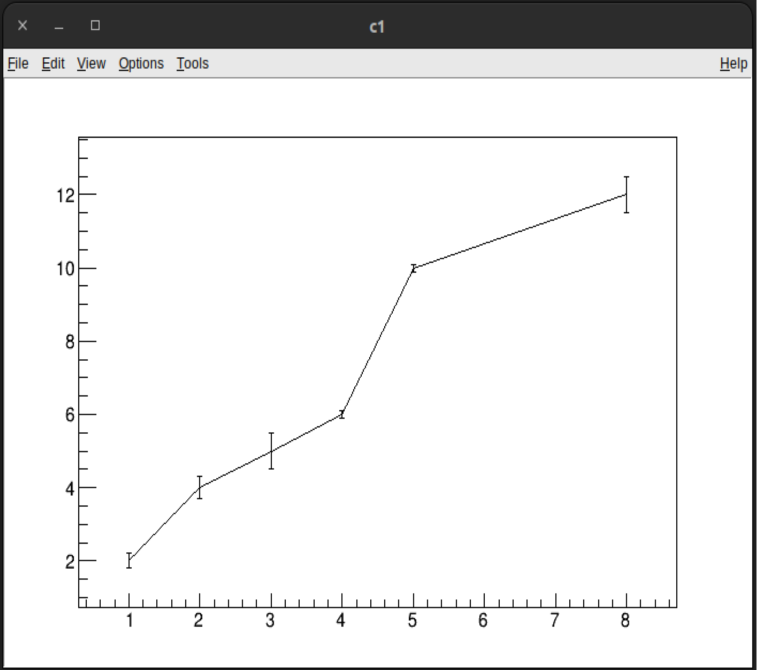


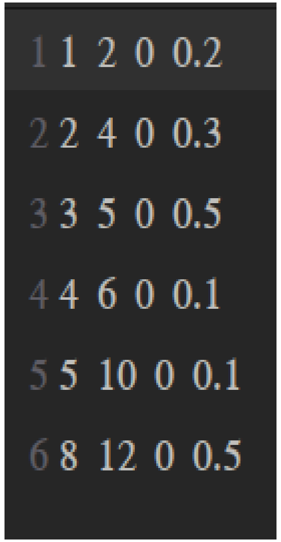
```
4  2  0.2  0.03
2  4  0.2  0.05
3  6  0.3  0.03
5  12  0.6  0.05
6  24  0.5  0.04
4  28  0.9  0.07
8  35  0.2  0.01
9  26  0.8  0.04
10  25  0.5  0.06
```

Vertical Error Bars



```
void errgr()
{
TGraphErrors *gr = new TGraphErrors();

fstream file;
file.open("data_error.txt", ios::in);

double x, y, ex, ey;
int n=6;
while(1)
{
file >> x >> y >> ex >> ey;
n=gr->GetN();
gr->SetPoint(n,x,y);
gr->SetPointError(n,ex,ey);
if(file.eof()) break;
}
gr->Draw();
}
```

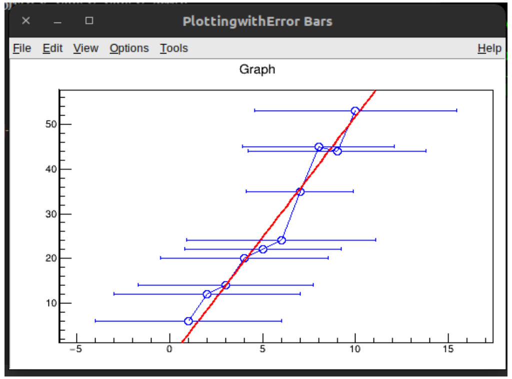


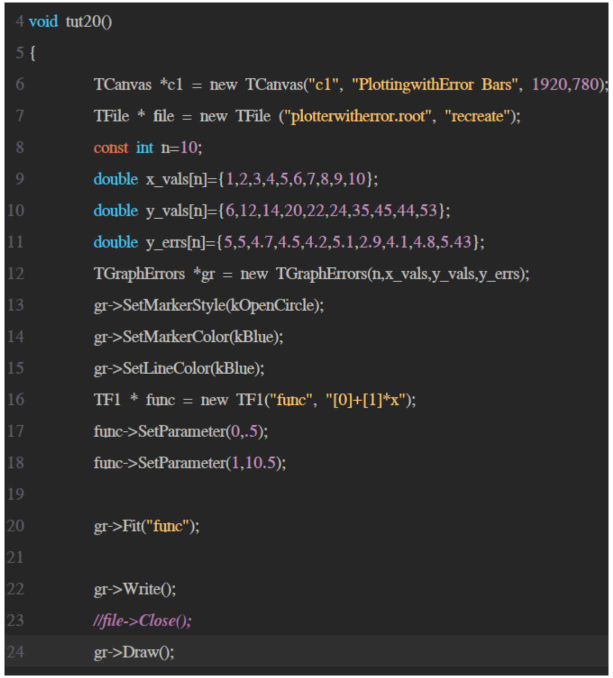


```
1 2 0 0.2
2 4 0 0.3
3 5 0 0.5
4 6 0 0.1
5 10 0 0.1
8 12 0 0.5
```

User Defined Functions

- Void Macro();
- {
- Draw a canvas;
- Create a function and give the arguments like name of the function, function equation and range of the function;
- Set Parameters for function variables
- Draw the function
- }



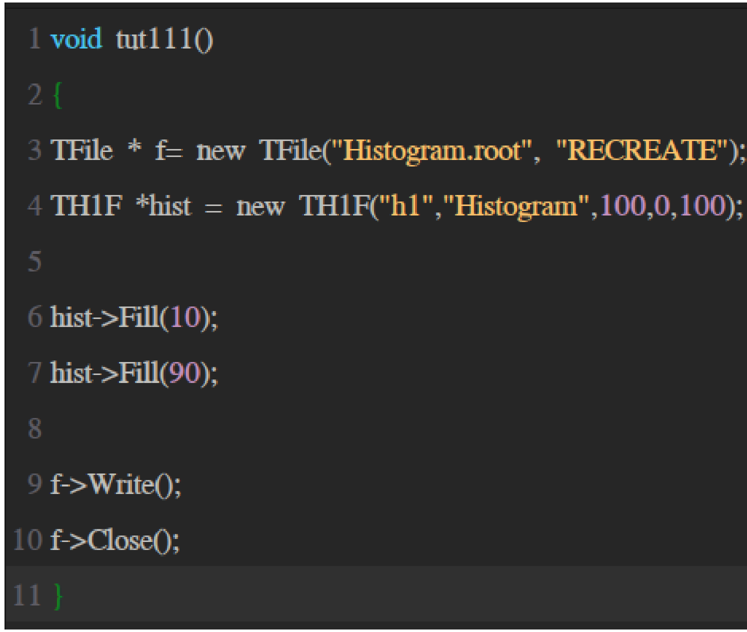


```
void tut20()
{
        TCanvas *c1 = new TCanvas("c1", "PlottingwithError Bars", 1920,780);
        TFile * file = new TFile ("plotterwitherror.root", "recreate");
        const int n=10;
        double x_vals[n]={1,2,3,4,5,6,7,8,9,10};
        double y_vals[n]={6,12,14,20,22,24,35,45,44,53};
        double y_errs[n]={5,5,4.7,4.5,4.2,5.1,2.9,4.1,4.8,5.43};
        TGraphErrors *gr = new TGraphErrors(n,x_vals,y_vals,y_errs);
        gr->SetMarkerStyle(kOpenCircle);
        gr->SetMarkerColor(kBlue);
        gr->SetLineColor(kBlue);
        TF1 * func = new TF1("func", "[0]+[1]*x");
        func->SetParameter(0,.5);
        func->SetParameter(1,10.5);

        gr->Fit("func");

        gr->Write();
        //file->Close();
        gr->Draw();
```

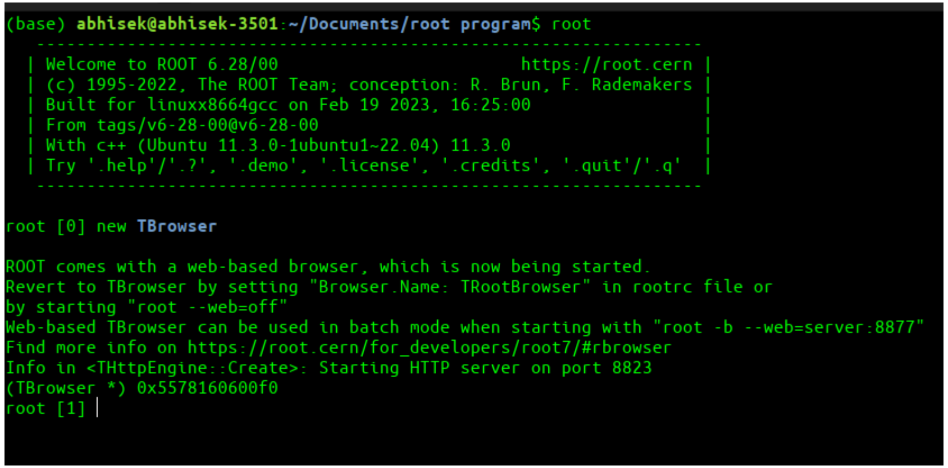
TFile

- TFile is a class which is used to read ,write and manipulate root files which are binary files used to storage large amounts of data in a structured way.
- With TFile, users can create new ROOT files, open existing files, and browse the contents of a file. It also provides methods for reading and writing data to and from ROOT files.

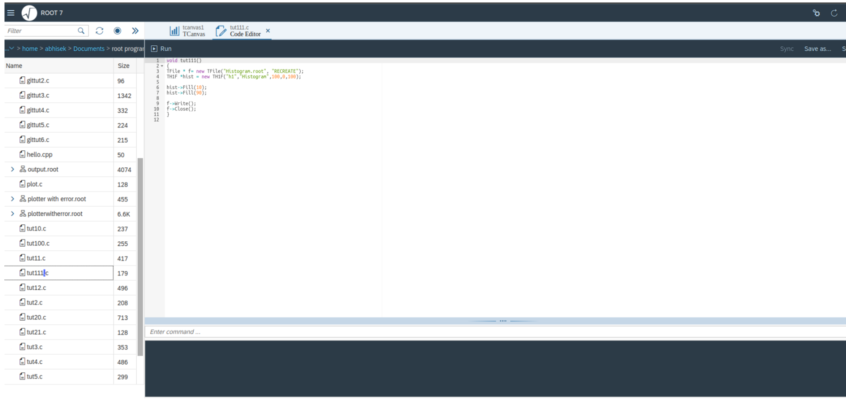
Storing and Loading an object In a TFile



```
void tut111()
{
TFile * f= new TFile("Histogram.root", "RECREATE");
TH1F *hist = new TH1F("h1","Histogram",100,0,100);

hist->Fill(10);
hist->Fill(90);

f->Write();
f->Close();
}
```



```
(base) abhisek@abhisek-3501:~/Documents/root program$ root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.28/00                        https://root.cern |
  | (c) 1995-2022, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Feb 19 2023, 16:25:00                 |
  | From tags/v6-28-00@v6-28-00                                      |
  | With c++ (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0                   |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------

root [0] new TBrowser

ROOT comes with a web-based browser, which is now being started.
Revert to TBrowser by setting "Browser.Name: TRootBrowser" in rootrc file or
by starting "root --web=off"
Web-based TBrowser can be used in batch mode when starting with "root -b --web=server:8877"
Find more info on https://root.cern/for_developers/root7/#rbrowser
Info in <THttpEngine::Create>: Starting HTTP server on port 8823
(TBrowser *) 0x5578160600f0
root [1] |
```



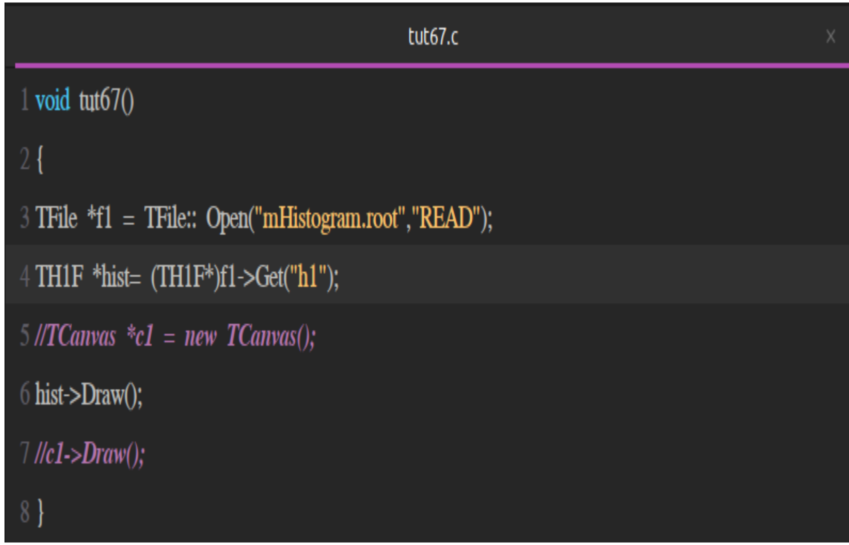
```
... > home > abhisek > Documents > root progra

Name                      Size
gittut2.c                 96
gittut3.c                 1342
gittut4.c                 332
gittut5.c                 224
gittut6.c                 215
hello.cpp                 50
> output.root             4074
plot.c                    128
> plotter with error.root 455
> plotterwitherror.root   6.6K
tut10.c                   237
tut100.c                  255
tut11.c                   417
tut111.c                  179
tut12.c                   496
tut2.c                    208
tut20.c                   713
tut21.c                   128
tut3.c                    353
tut4.c                    486
tut5.c                    299
```

```
void tut111()
{
TFile * f= new TFile("Histogram.root", "RECREATE");
TH1F *hist = new TH1F("h1","Histogram",100,0,100);

hist->Fill(10);
hist->Fill(90);

f->Write();
f->Close();
}
```

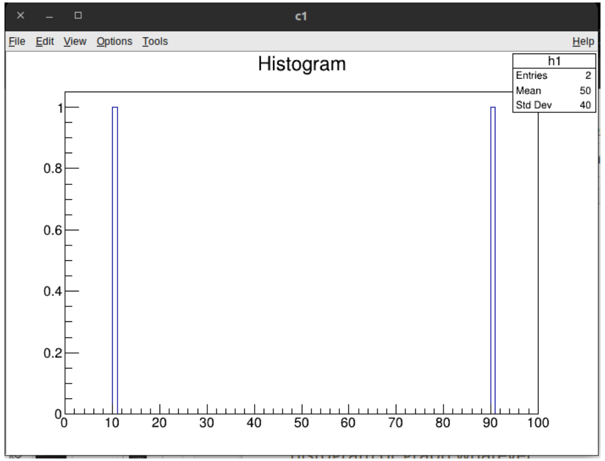
Reading a File

We have to call the file by using the TFile class.

Next we have to call the histogram or graph whatever inside that file using the preferred TH1F or TH1D.


tut67.c

```
void tut67()
{
TFile *f1 = TFile:: Open("mHistogram.root","READ");
TH1F *hist= (TH1F*)f1->Get("h1");
//TCanvas *c1 = new TCanvas();
hist->Draw();
//c1->Draw();
}
```

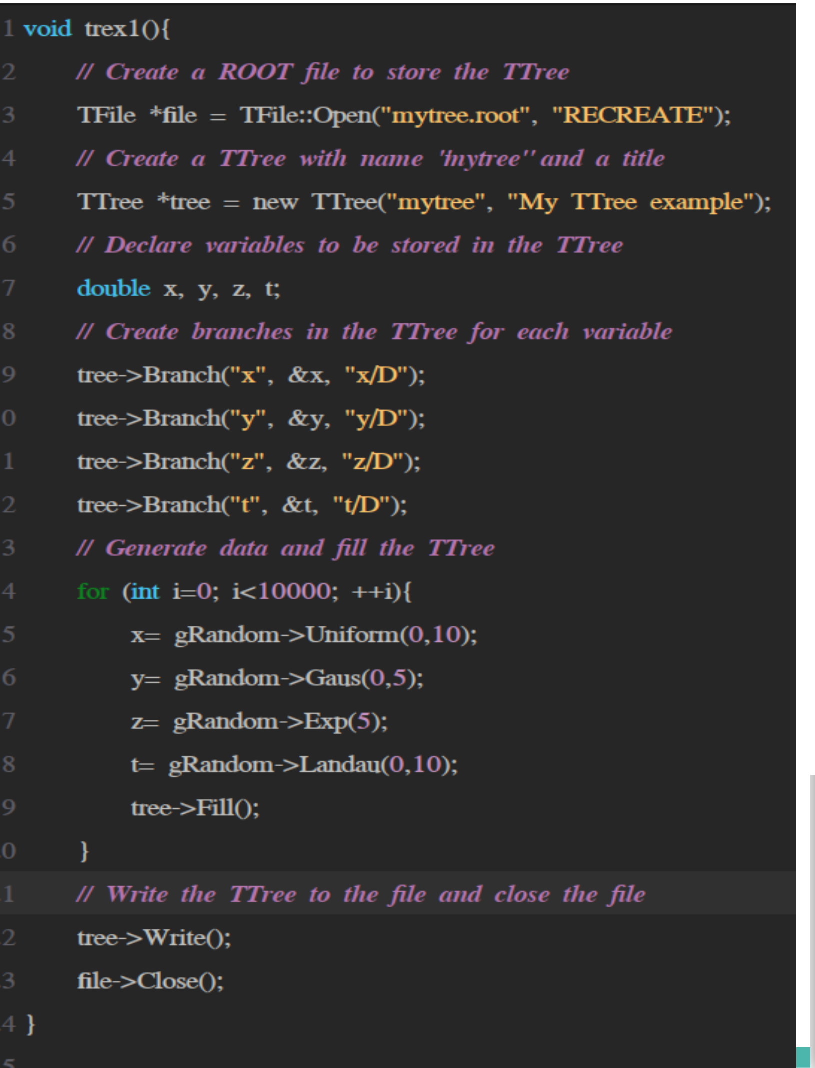


TTree and storing data in a TTree

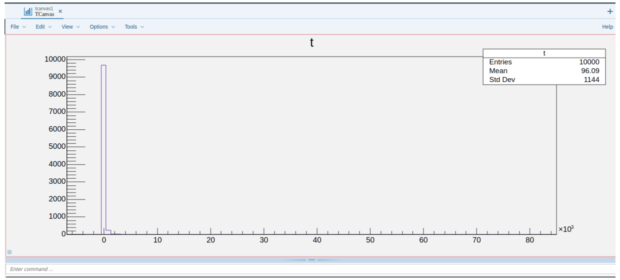
In the ROOT data analysis framework, TTree is a data structure used for storing and analyzing large sets of data. It is a tree-like structure, where each node represents an event or an entry in the dataset, and each branch represents a variable or a set of variables associated with that event/entry.

Create a some Data and put it In a TTree

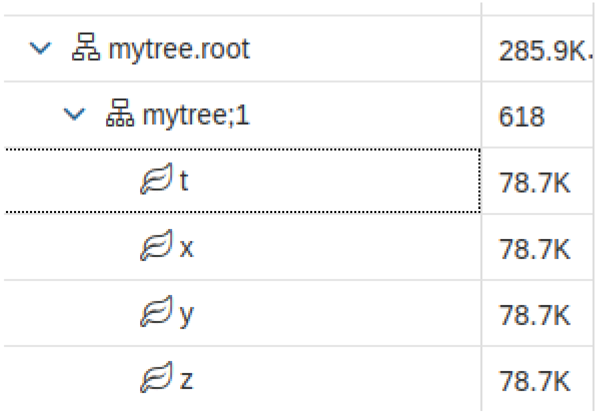
- First We will create a root file using TFile class where we will be saving our tree file.
- Now we will create a tree using TTree Class.
- Then We will create some variables and put these variables in the branches of this tree.
- Fill these variables.
- And at the end we will write these data into tree and close the file.



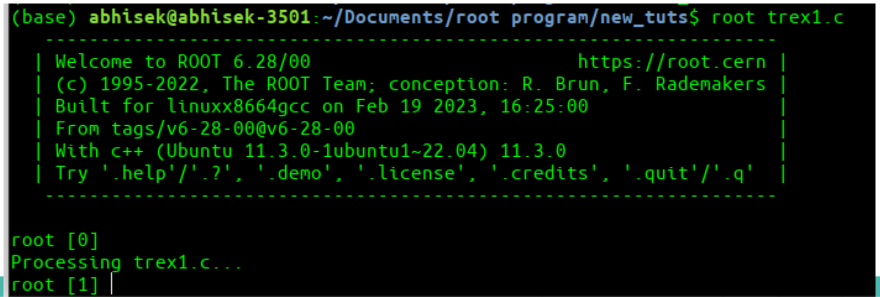
```cpp
void trex1(){
    // Create a ROOT file to store the TTree
    TFile *file = TFile::Open("mytree.root", "RECREATE");
    // Create a TTree with name 'mytree'' and a title
    TTree *tree = new TTree("mytree", "My TTree example");
    // Declare variables to be stored in the TTree
    double x, y, z, t;
    // Create branches in the TTree for each variable
    tree->Branch("x", &x, "x/D");
    tree->Branch("y", &y, "y/D");
    tree->Branch("z", &z, "z/D");
    tree->Branch("t", &t, "t/D");
    // Generate data and fill the TTree
    for (int i=0; i<10000; ++i){
        x= gRandom->Uniform(0,10);
        y= gRandom->Gaus(0,5);
        z= gRandom->Exp(5);
        t= gRandom->Landau(0,10);
        tree->Fill();
    }
    // Write the TTree to the file and close the file
    tree->Write();
    file->Close();
}
```



```
t
Entries   10000
Mean      96.09
Std Dev    1144
```



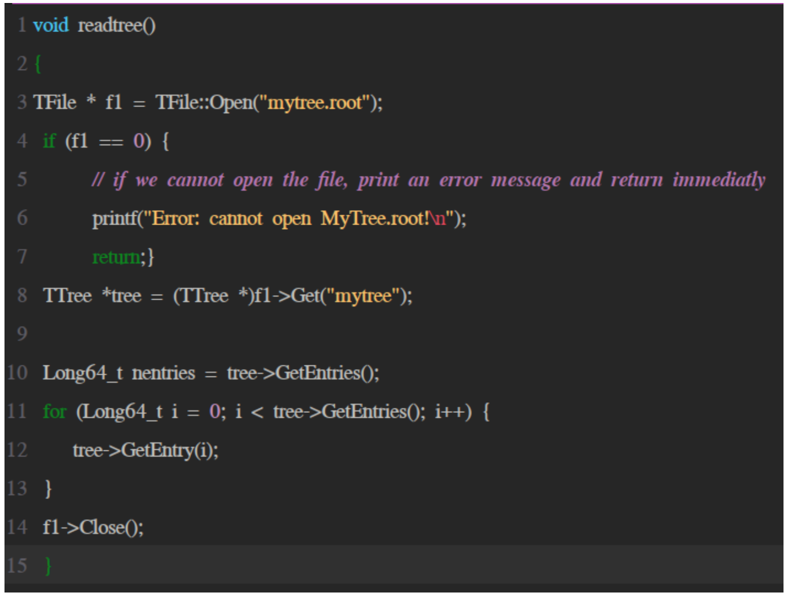
mytree.root	285.9K
mytree;1	618
t	78.7K
x	78.7K
y	78.7K
z	78.7K



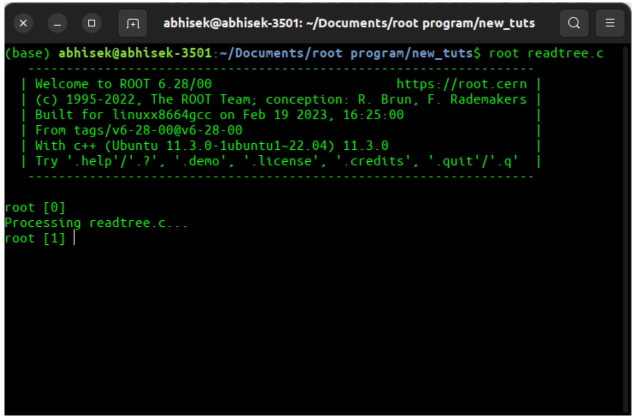
```
(base) abhisek@abhisek-3501:~/Documents/root program/new_tuts$ root trex1.c
   ------------------------------------------------------------------
  | Welcome to ROOT 6.28/00                        https://root.cern |
  | (c) 1995-2022, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Feb 19 2023, 16:25:00                 |
  | From tags/v6-28-00@v6-28-00                                      |
  | With c++ (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0                   |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------

root [0]
Processing trex1.c...
root [1]
```

Reading Data from TTree

- Like TFile we will use TTree get function to read our preferred tree.
- Call the File to read.
- Call the tree using Get function.
- Loop over entries and that’s it we will get our tree in TBrowser.



```
void readtree()
{
TFile * f1 = TFile::Open("mytree.root");
 if (f1 == 0) {
        // if we cannot open the file, print an error message and return immediatly
        printf("Error: cannot open MyTree.root!\n");
        return;}
 TTree *tree = (TTree *)f1->Get("mytree");

 Long64_t nentries = tree->GetEntries();
 for (Long64_t i = 0; i < tree->GetEntries(); i++) {
     tree->GetEntry(i);
 }
 f1->Close();
 }
```



```
(base) abhisek@abhisek-3501:~/Documents/root program/new_tuts$ root readtree.c
   ------------------------------------------------------------------
  | Welcome to ROOT 6.28/00                        https://root.cern |
  | (c) 1995-2022, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Feb 19 2023, 16:25:00                 |
  | From tags/v6-28-00@v6-28-00                                      |
  | With c++ (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0                   |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------

root [0]
Processing readtree.c...
root [1]
```

THANK YOU