4-й семестр, lesson 7, Reading and Writing NTuples

- Corrections to previous exercise, Reading:
- 1) in Header section
- static int Itape;
- static int Ispln;
- static int Ievtn;
- static TBranch * b_Itape;
- static TBranch * b_Ispln;
- static TBranch * b_Ievtn;

Corrections for reading

- 2) static TBranch * b_Itape;
- static TBranch * b_Ispln;
- static TBranch * b_Ievtn;
- 3) after opening of input Ntuple, but before the Event Loop:
- fChain->SetBranchStatus("Itape", 1);
- fChain->SetBranchStatus("Ispln", 1);
- fChain->SetBranchStatus("Ievtn", 1);
- 4) fChain->SetBranchAddress("Itape", &Itape, &b_Itape);
- fChain->SetBranchAddress("Ispln", &Ispln, &b_Ispln);
- fChain->SetBranchAddress("Ievtn", &Ievtn, &b_Ievtn);
- 5) Itape = 0; Ispln = 0; Ievtn = 0;

Sequence for Writing

- 1) in Header section:
- static int nrun_t3;
- static int nspl_t3;
- static int nevt_t3;
- static double m3pi;
- 2) before the Event Loop
- TFile *newfile = new TFile("small.root","recreate");
- t3ev= new TTree("t3ev", " short ntuple ");

Writing NTuple

- 3) t3ev->Branch("nrun_t3", &nrun_t3, "nrun_t3/I");
- t3ev->Branch("nspl_t3", &nspl_t3, "nspl_t3/I");
- t3ev->Branch("nevt_t3", &nevt_t3, "nevt_t3/I");
- t3ev->Branch("m3pi_t3", &m3pi_t3, "m3pi_t3/D");
- 4) nrun_t3 = 0;
- nspl_t3 = 0;
- nevt_t3 = 0;
- m3pi_t3 = 0;

Writing NTuple

- 5)In Event Loop, fill variables
- nrun_t3 = Itape;
- nspl_t3 = Ispln;
- nevt_t3 = Ievtn;
- m3pi_t3 = fm3;
- 6) and copy the event data to new Ntuple
- t3ev -> Fill();
- 7) After Event Loop
- t3ev->AutoSave();
- newfile->Close();

Complete previous exercise – produce new Tree

- Starting from previous example in /nfs/lfi.mipt.su/data/nikola/ves/run42/example_4_edited.C
- Produce new tree «omega» with 4 branches:
- Itape, Ispln, Ievtn, m(pi+pi-pi0)
- Useful comand : l9024->Print(), it helps to find the type of objects
- Please do not use the identifiers from input Tree for definition of new structure.
- Write new Tree to file.

MakeClass from produced file

- How to prepare reading of produced file « small.root»
- There is Tree called «t3ev» in this file
- Needed the following command file called setMakeClass.C :
- {
- TFile *_file0 = TFile::Open("small.root");
- TTree * new_tree = (TTree*)gROOT->FindObject("t3ev");
- new_tree->MakeClass("read_tree");
- }
- Then in ROOT :
- .X setMakeClass.C
- Files: read_tree.h and read_tree.C generated from TTree: t3ev

Look for produced read_tree.h and read_tree.h

- Created class read_tree with different methods and statements like
- 1) Double_t m3pi_t3;
- 2) TBranch *b_m3pi_t3;
- 3) fChain->SetBranchAddress("m3pi_t3", &m3pi_t3, &b_m3pi_t3);
- In ROOT section user can do:
- 1) compilation, Root > .L read_tree.C
- 2) declare class Root> read_tree t;
- 3) open input file, find input Tree «input» and call t.Loop(input) with

An example of reading sequence

- root -l
- TFile *_file0 = TFile::Open("small.root");
- TTree * input_tree = (TTree*)gROOT->FindObject("t3ev"   );
- .L read_tree.C
- read_tree t;
- t.Init(input_tree);
- t.Loop( );

- The input Tree name is written in read_tree.h by MakeClass
- If the same Tree is used in reading, then t.Init(…) is not needed

Work within read_tree.C

- For example, user can declare a histogram, book it (before the Loop call), fill it in the Loop. Then after the Loop user can open an output file, write the produce histogram and close the output file. An example of corresponding statements :
- 1) TH1D * hist_m3pi;
- 2) hist_m3pi = new TH1D("hist_m3pi", "hist_m3pi", Nbins, Xlow, Xhigh);
- 3) double value = … ;
- hist_m3pi->Fill(value);
- 4) TFile *_filewOut= TFile::Open("hist_file.root", "recreate");
- hist_m3pi->Write();
- _filewOut->Close();

- Run the t.Loop() (see previous page)

Work with hist_file.root (resonances in ($\pi^+ \pi^- \pi^0$) system

- root –l hist_file.root
- .ls
- hist_m3pi -> Draw();

- Try to fit the distribution like the fit hist_m3pi by a Gauss function in mass range (0.65, 0.90) GeV like the example in /nfs/lfi.mipt.su/data/nikola/ves/run42/example_6_edited.C :
- {
- TFile *_file0 = TFile::Open("hist_file.root");
- TF1 *myfit2 = new TF1("myfit2", "gaus(0)",0.65, 0.9);
- hist_m3pi ->Draw();
- gStyle-> SetOptFit();
- //myfit2 -> SetParameter(0, xxx.);
- //myfit2 -> SetParameter(1, x.x);
- //myfit2 -> SetParameter(2, xxxx0);
- hist_m3pi->Fit("myfit2","eR+");
- }
- needed edition: please set the starting values for 3 parameters in the Gauss

Exercises cont.

- Please compare the fitted curve with histogram.
- 3) In order to improve agreement between data and fit result, let's include a Background term (linear function). An example available in /nfs/lfi.mipt.su/data/nikola/ves/run42/example_7_edited.C{
- TFile *_file0 = TFile::Open("hist_file.root");
- TF1 *myfit5 = new TF1("myfit5","gaus(0)+pol1(3)",0.65, 0.9);
- hist_m3pi->Draw();
- gStyle-> SetOptFit() ;
- //myfit5 -> SetParameter(0, xxxx);
- //myfit5-> SetParameter(1, xxx);
- //myfit5-> SetParameter(2, xxxxx);
- //myfit5-> SetParameter(3, xxxxx);
- //myfit5-> SetParameter(4, xxxxx);
- hist_m3pi->Fit("myfit5","eR+");
- }

Exercises cont.

- Parameters 0, 1, 2 are associated with Gauss
- Parameters 3,4 are associated with the linear fubction (pol1).
- Again, needed a starting point for 3+2 parameters (which might be very approximate)
- Run the fit and look for result. Sttill the χ^2 is bad, but the agreement between the data and the fit result is significantly better.