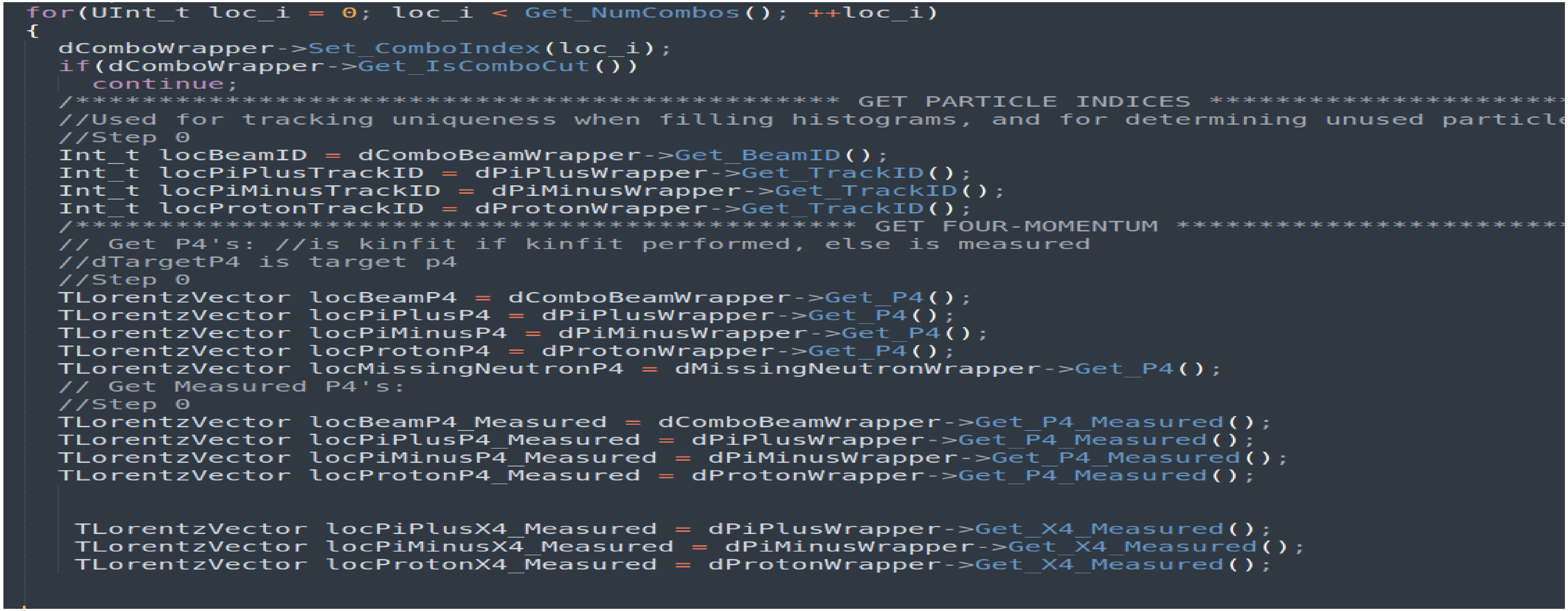
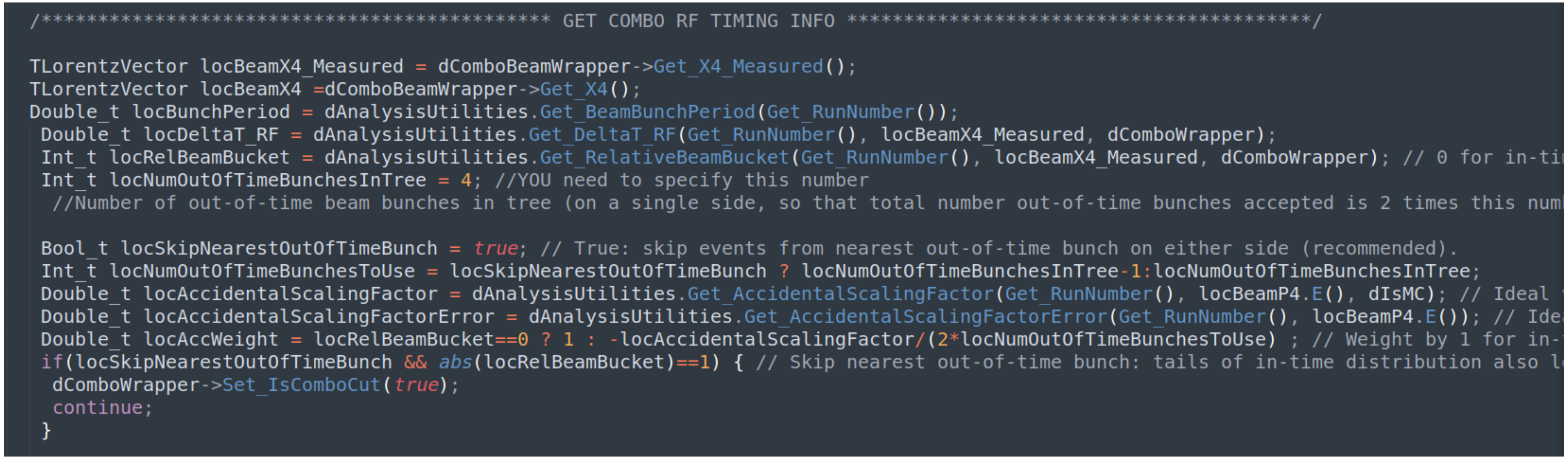


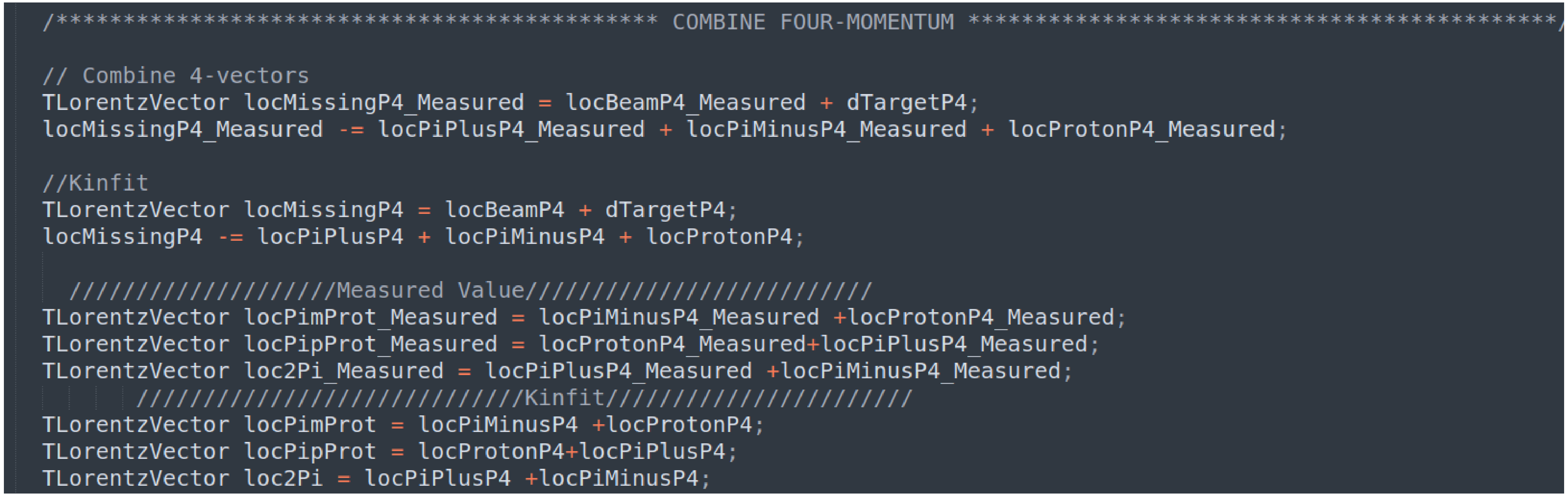
```
for(UInt_t loc_i = 0; loc_i < Get_NumCombos(); ++loc_i)
{
    dComboWrapper->Set_ComboIndex(loc_i);
    if(dComboWrapper->Get_IsComboCut())
        continue;
    /********************************************** GET PARTICLE INDICES ***************************
    //Used for tracking uniqueness when filling histograms, and for determining unused particle
    //Step 0
    Int_t locBeamID = dComboBeamWrapper->Get_BeamID();
    Int_t locPiPlusTrackID = dPiPlusWrapper->Get_TrackID();
    Int_t locPiMinusTrackID = dPiMinusWrapper->Get_TrackID();
    Int_t locProtonTrackID = dProtonWrapper->Get_TrackID();
    /*********************************************** GET FOUR-MOMENTUM ************************
    // Get P4's: //is kinfit if kinfit performed, else is measured
    //dTargetP4 is target p4
    //Step 0
    TLorentzVector locBeamP4 = dComboBeamWrapper->Get_P4();
    TLorentzVector locPiPlusP4 = dPiPlusWrapper->Get_P4();
    TLorentzVector locPiMinusP4 = dPiMinusWrapper->Get_P4();
    TLorentzVector locProtonP4 = dProtonWrapper->Get_P4();
    TLorentzVector locMissingNeutronP4 = dMissingNeutronWrapper->Get_P4();
    // Get Measured P4's:
    //Step 0
    TLorentzVector locBeamP4_Measured = dComboBeamWrapper->Get_P4_Measured();
    TLorentzVector locPiPlusP4_Measured = dPiPlusWrapper->Get_P4_Measured();
    TLorentzVector locPiMinusP4_Measured = dPiMinusWrapper->Get_P4_Measured();
    TLorentzVector locProtonP4_Measured = dProtonWrapper->Get_P4_Measured();


     TLorentzVector locPiPlusX4_Measured = dPiPlusWrapper->Get_X4_Measured();
     TLorentzVector locPiMinusX4_Measured = dPiMinusWrapper->Get_X4_Measured();
     TLorentzVector locProtonX4_Measured = dProtonWrapper->Get_X4_Measured();
```

Accidental Subtraction



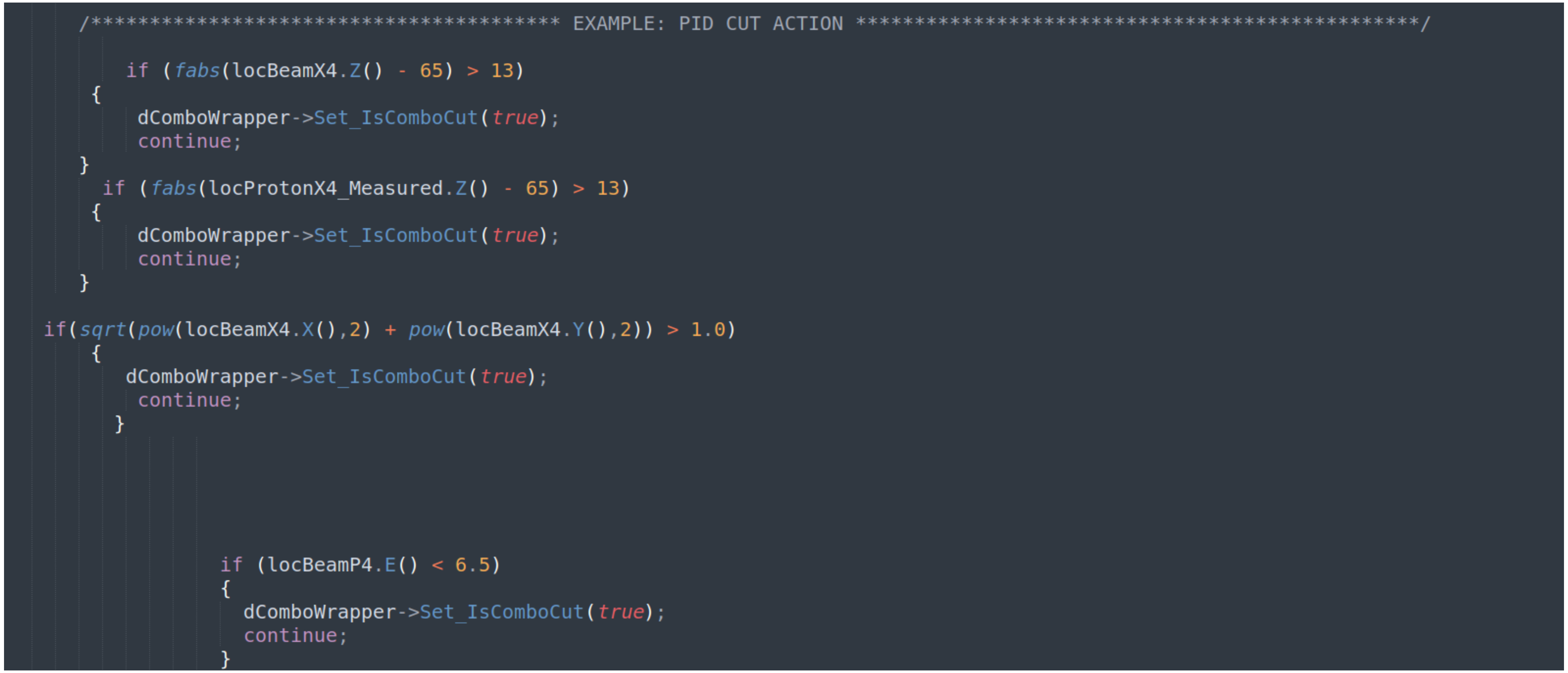
```
/******************************************** GET COMBO RF TIMING INFO ****************************************/

TLorentzVector locBeamX4_Measured = dComboBeamWrapper->Get_X4_Measured();
TLorentzVector locBeamX4 =dComboBeamWrapper->Get_X4();
Double_t locBunchPeriod = dAnalysisUtilities.Get_BeamBunchPeriod(Get_RunNumber());
 Double_t locDeltaT_RF = dAnalysisUtilities.Get_DeltaT_RF(Get_RunNumber(), locBeamX4_Measured, dComboWrapper);
 Int_t locRelBeamBucket = dAnalysisUtilities.Get_RelativeBeamBucket(Get_RunNumber(), locBeamX4_Measured, dComboWrapper); // 0 for in-ti
 Int_t locNumOutOfTimeBunchesInTree = 4; //YOU need to specify this number
  //Number of out-of-time beam bunches in tree (on a single side, so that total number out-of-time bunches accepted is 2 times this num

 Bool_t locSkipNearestOutOfTimeBunch = true; // True: skip events from nearest out-of-time bunch on either side (recommended).
 Int_t locNumOutOfTimeBunchesToUse = locSkipNearestOutOfTimeBunch ? locNumOutOfTimeBunchesInTree-1:locNumOutOfTimeBunchesInTree;
 Double_t locAccidentalScalingFactor = dAnalysisUtilities.Get_AccidentalScalingFactor(Get_RunNumber(), locBeamP4.E(), dIsMC); // Ideal
 Double_t locAccidentalScalingFactorError = dAnalysisUtilities.Get_AccidentalScalingFactorError(Get_RunNumber(), locBeamP4.E()); // Ide
 Double_t locAccWeight = locRelBeamBucket==0 ? 1 : -locAccidentalScalingFactor/(2*locNumOutOfTimeBunchesToUse) ; // Weight by 1 for in-
 if(locSkipNearestOutOfTimeBunch && abs(locRelBeamBucket)==1) { // Skip nearest out-of-time bunch: tails of in-time distribution also l
  dComboWrapper->Set_IsComboCut(true);
  continue;
 }
```

Defining Variable of our interest



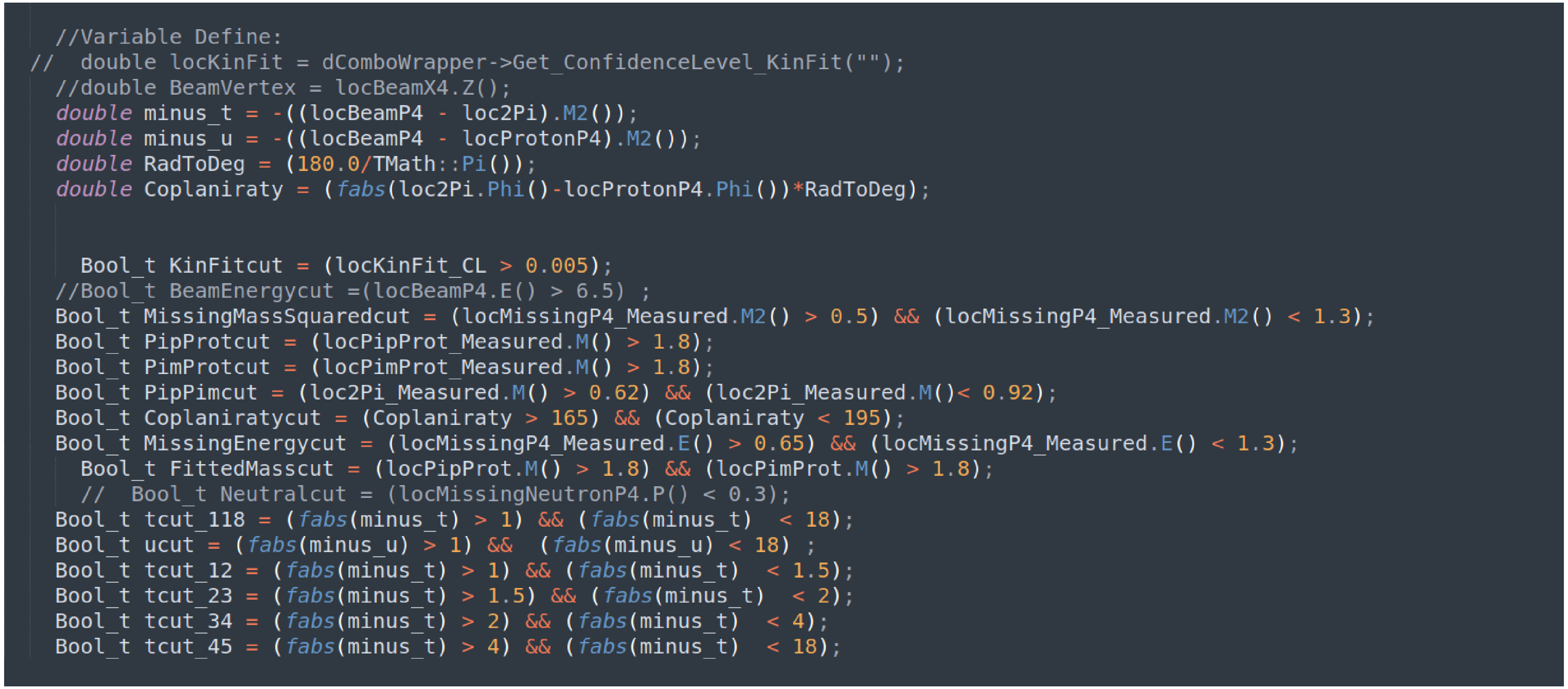
```
/******************************************** COMBINE FOUR-MOMENTUM ********************************************/

// Combine 4-vectors
TLorentzVector locMissingP4_Measured = locBeamP4_Measured + dTargetP4;
locMissingP4_Measured -= locPiPlusP4_Measured + locPiMinusP4_Measured + locProtonP4_Measured;

//Kinfit
TLorentzVector locMissingP4 = locBeamP4 + dTargetP4;
locMissingP4 -= locPiPlusP4 + locPiMinusP4 + locProtonP4;

  ///////////////////Measured Value/////////////////////////
TLorentzVector locPimProt_Measured = locPiMinusP4_Measured +locProtonP4_Measured;
TLorentzVector locPipProt_Measured = locProtonP4_Measured+locPiPlusP4_Measured;
TLorentzVector loc2Pi_Measured = locPiPlusP4_Measured +locPiMinusP4_Measured;
       ////////////////////////////Kinfit/////////////////////
TLorentzVector locPimProt = locPiMinusP4 +locProtonP4;
TLorentzVector locPipProt = locProtonP4+locPiPlusP4;
TLorentzVector loc2Pi = locPiPlusP4 +locPiMinusP4;
```

Applying Vertex and Energy Cut



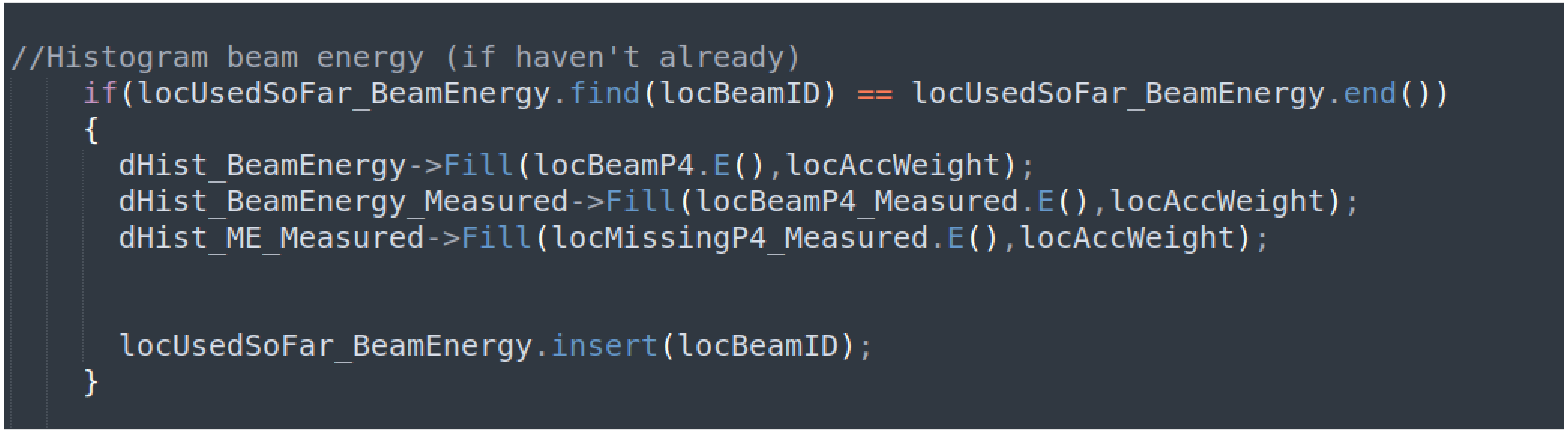
```
    /**************************************** EXAMPLE: PID CUT ACTION ************************************************/

        if (fabs(locBeamX4.Z() - 65) > 13)
      {
         dComboWrapper->Set_IsComboCut(true);
         continue;
    }
       if (fabs(locProtonX4_Measured.Z() - 65) > 13)
      {
         dComboWrapper->Set_IsComboCut(true);
         continue;
    }

 if(sqrt(pow(locBeamX4.X(),2) + pow(locBeamX4.Y(),2)) > 1.0)
      {
        dComboWrapper->Set_IsComboCut(true);
         continue;
       }




            if (locBeamP4.E() < 6.5)
            {
              dComboWrapper->Set_IsComboCut(true);
              continue;
            }
```

Defining variable and cuts



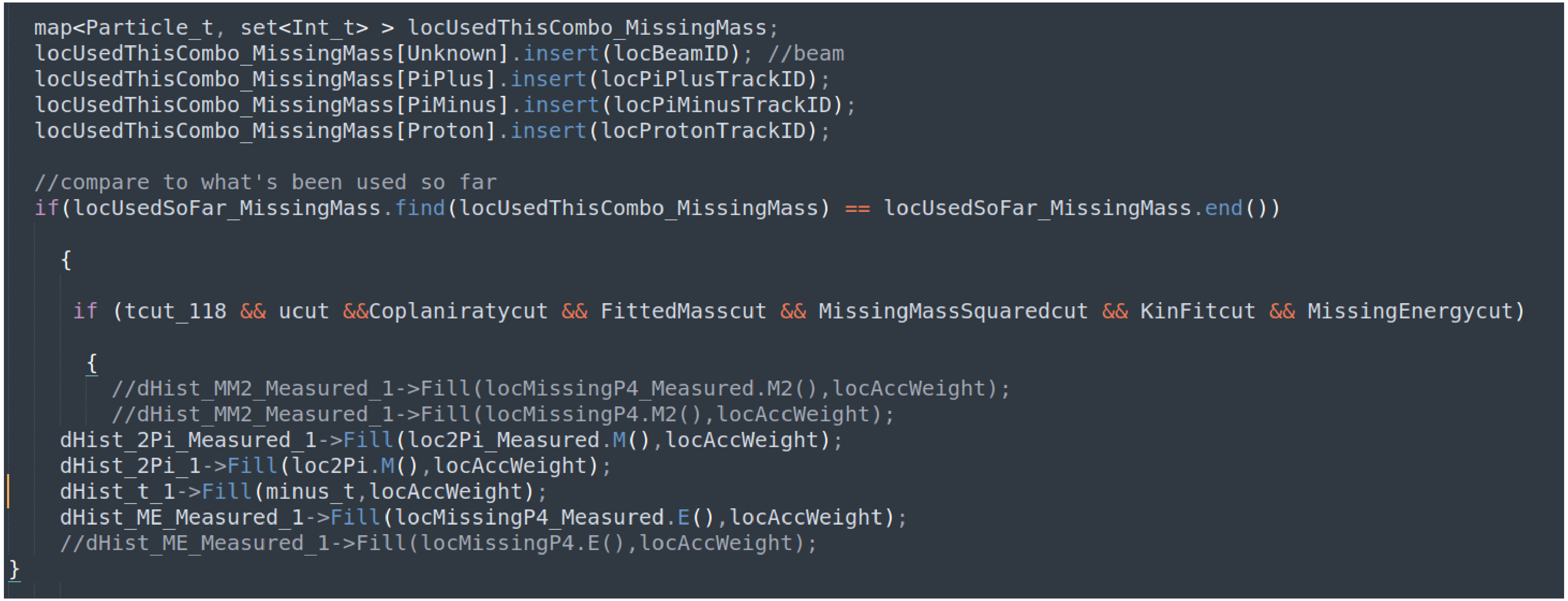
```
  //Variable Define:
//  double locKinFit = dComboWrapper->Get_ConfidenceLevel_KinFit("");
  //double BeamVertex = locBeamX4.Z();
  double minus_t = -((locBeamP4 - loc2Pi).M2());
  double minus_u = -((locBeamP4 - locProtonP4).M2());
  double RadToDeg = (180.0/TMath::Pi());
  double Coplaniraty = (fabs(loc2Pi.Phi()-locProtonP4.Phi())*RadToDeg);


    Bool_t KinFitcut = (locKinFit_CL > 0.005);
  //Bool_t BeamEnergycut =(locBeamP4.E() > 6.5) ;
  Bool_t MissingMassSquaredcut = (locMissingP4_Measured.M2() > 0.5) && (locMissingP4_Measured.M2() < 1.3);
  Bool_t PipProtcut = (locPipProt_Measured.M() > 1.8);
  Bool_t PimProtcut = (locPimProt_Measured.M() > 1.8);
  Bool_t PipPimcut = (loc2Pi_Measured.M() > 0.62) && (loc2Pi_Measured.M()< 0.92);
  Bool_t Coplaniratycut = (Coplaniraty > 165) && (Coplaniraty < 195);
  Bool_t MissingEnergycut = (locMissingP4_Measured.E() > 0.65) && (locMissingP4_Measured.E() < 1.3);
    Bool_t FittedMasscut = (locPipProt.M() > 1.8) && (locPimProt.M() > 1.8);
    //  Bool_t Neutralcut = (locMissingNeutronP4.P() < 0.3);
  Bool_t tcut_118 = (fabs(minus_t) > 1) && (fabs(minus_t)  < 18);
  Bool_t ucut = (fabs(minus_u) > 1) &&  (fabs(minus_u) < 18) ;
  Bool_t tcut_12 = (fabs(minus_t) > 1) && (fabs(minus_t)  < 1.5);
  Bool_t tcut_23 = (fabs(minus_t) > 1.5) && (fabs(minus_t)  < 2);
  Bool_t tcut_34 = (fabs(minus_t) > 2) && (fabs(minus_t)  < 4);
  Bool_t tcut_45 = (fabs(minus_t) > 4) && (fabs(minus_t)  < 18);
```

Filling Histogram



```
//Histogram beam energy (if haven't already)
    if(locUsedSoFar_BeamEnergy.find(locBeamID) == locUsedSoFar_BeamEnergy.end())
    {
      dHist_BeamEnergy->Fill(locBeamP4.E(),locAccWeight);
      dHist_BeamEnergy_Measured->Fill(locBeamP4_Measured.E(),locAccWeight);
      dHist_ME_Measured->Fill(locMissingP4_Measured.E(),locAccWeight);


      locUsedSoFar_BeamEnergy.insert(locBeamID);
    }
```

Filling Invariant Mass and |t| distribution



```
  map<Particle_t, set<Int_t> > locUsedThisCombo_MissingMass;
  locUsedThisCombo_MissingMass[Unknown].insert(locBeamID); //beam
  locUsedThisCombo_MissingMass[PiPlus].insert(locPiPlusTrackID);
  locUsedThisCombo_MissingMass[PiMinus].insert(locPiMinusTrackID);
  locUsedThisCombo_MissingMass[Proton].insert(locProtonTrackID);

  //compare to what's been used so far
  if(locUsedSoFar_MissingMass.find(locUsedThisCombo_MissingMass) == locUsedSoFar_MissingMass.end())

    {

     if (tcut_118 && ucut &&Coplaniratycut && FittedMasscut && MissingMassSquaredcut && KinFitcut && MissingEnergycut)

      {
        //dHist_MM2_Measured_1->Fill(locMissingP4_Measured.M2(),locAccWeight);
        //dHist_MM2_Measured_1->Fill(locMissingP4.M2(),locAccWeight);
    dHist_2Pi_Measured_1->Fill(loc2Pi_Measured.M(),locAccWeight);
    dHist_2Pi_1->Fill(loc2Pi.M(),locAccWeight);
    dHist_t_1->Fill(minus_t,locAccWeight);
    dHist_ME_Measured_1->Fill(locMissingP4_Measured.E(),locAccWeight);
    //dHist_ME_Measured_1->Fill(locMissingP4.E(),locAccWeight);
}
```