```
#ifndef Track_h
#define Track_h 1
////////////////////////////////////////////////////////////////////////////////
//
#include "globals.hh"
#include <vector>

#include "G4VHit.hh"
#include "G4THitsCollection.hh"
#include "G4Allocator.hh"
#include "G4ThreeVector.hh"

#include "G4HCofThisEvent.hh"

////////////////////////////////////////////////////////////////////////////////
//
class Track : public G4VHit
{
public:
  Track ();
  Track (G4int aTrackId, G4int aPDG,
         G4ThreeVector aPosition, G4ThreeVector aMomentumDirection, G4double the
KinEnergy,
         G4int aMotherTrackId,G4String aVertexVolumeName,G4String aCreatorProces
sName);

  ~Track ();
  Track (const Track&);
  const Track& operator= (const Track&);
  int operator== (const Track&) const;

  inline void* operator new(size_t);
  inline void  operator delete(void*);

  inline G4int          GetTrackId(){return theTrackId;}
  inline G4int          GetPDGCode(){return thePDG;}
  inline G4ThreeVector  GetPosition(){return thePosition;}
  inline G4double       GetKinEnergy(){return theKinEnergy;}
  inline G4ThreeVector  GetMomentumDirection(){return theMomentumDirection;}
  inline G4int          GetMotherTrackId(){return theMotherTrackId;}
  inline G4String       GetVertexVolumeName(){return theVertexVolumeName;}
  inline G4String       GetCreatorProcessName(){return theCreatorProcessName;}

  void Draw () {};
  void Print () {};
  void clear () {};
  void DrawAll () {};
  void PrintAll () {};

private:
  G4int          theTrackId;
  G4int          thePDG;              //Using this code we can in ROOT to the bu
ilt in table and create a
                                      //TParticlePDG that contain particleMass p
articleCharge
  G4ThreeVector  thePosition;
  G4ThreeVector  theMomentumDirection;
  G4double       theKinEnergy;
  G4int          theMotherTrackId;
  G4String       theVertexVolumeName;
  G4String       theCreatorProcessName;
};

typedef G4THitsCollection<Track> TracksCollection;

extern G4Allocator<Track> TrackAllocator;

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```



```
inline void* Track::operator new(size_t)
{
  void *aHit;
  aHit = (void *) TrackAllocator.MallocSingle();
  return aHit;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

inline void Track::operator delete(void *aHit)
{
  TrackAllocator.FreeSingle((Track*) aHit);
}


////////////////////////////////////////////////////////////////////////////////
#endif
```