./Exercise1/swapper.cpp Tue Oct 01 14:43:21 2024 1

```cpp
#include <iostream>
using namespace std;

// passed by reference
void swaper(int &a, int &b)
{
    int c;
    c = a;
    a = b;
    b = c;
}

int main()
{
    int i;
    int A[10] = {0,1,2,3,4,5,6,7,8,9};
    int B[10] = {10,11,12,13,14,15,16,17,18,19};

    cout << "Before\n";
    cout << "A = [";
    for(int idx : A)
        cout << idx << ",";
    cout << "]\n";

    cout << "B = [";
    for(int idx : B)
        cout << idx << ",";
    cout << "]\n";

    for (i = 0; i <= 9; i++)
    {
        swaper(A[i], B[i]);
    }

    cout << "After\n";
    cout << "A = [";
    for(int idx : A)
        cout << idx << ",";
    cout << "]\n";

    cout << "B = [";
    for(int idx : B)
        cout << idx << ",";
    cout << "]\n";
}
```

```cpp
#include <iostream>
#include <string>
using namespace std;

struct Students
{
    string name;
    string email;
    string username;
    string research;
};

void CoutStudent(const Students &s)
{
    cout << "Name: " << s.name << endl;
    cout << "Email: " << s.email << endl;
    cout << "Username: " << s.username << endl;
    cout << "Research: " << s.research << endl;
}

int main()
{
    string student_list[] = {"Rupesh", "Ameya", "Miranda"};

    Students rupesh, ameya, miranda;

    rupesh.name = "Rupesh Kannan";
    rupesh.email = "rupesh@wisc.edu";
    rupesh.username = "rupeshknn";
    rupesh.research = "Quantum Computing";

    ameya.name = "Ameya Thete";
    ameya.email = "ameya@wisc.edu";
    ameya.username = "ameyat05";
    ameya.research = "HEP";

    miranda.name = "Miranda Gorsuch";
    miranda.email = "miranda@wisc.edu";
    miranda.username = "mirandag12";
    miranda.research = "Cosmology";

    CoutStudent(rupesh);
    return 0;
}
```

```cpp
#include <iostream>
#include <string>
#include <map>

using namespace std;

int RockPaperScissor(string player1, string player2)
{
    map<string, int> rcpmap;

    // Insert some values into the map
    rcpmap["rock"] = 0;
    rcpmap["paper"] = 1;
    rcpmap["scissor"] = 2;

    if ((rcpmap[player1] + 1)%3 == rcpmap[player2])
        cout << "Player 2 wins!\n";
    else if ((rcpmap[player2] + 1)%3 == rcpmap[player1])
        cout << "Player 1 wins!\n";
    else if (player1 == player2)
        cout << "Draw!\n";
    else
        cout << "invalid input\n";

    return 0;
}

int main()
{
    string values[3] = {"rock", "paper", "scissor"};

    for (string c1: values){
        for (string c2: values){
            cout << "P1:" <<c1 << " P2:" << c2 << " ---> ";
            RockPaperScissor(c1, c2);
        }
    }
    return 0;
}
```

```
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>
#include <stdlib.h>
#include <string.h>

#include "t1.h"

#include <TMath.h>
#include <TFile.h>
#include <TTree.h>
#include <TH1F.h>
#include <TCanvas.h>
#include <TLorentzVector.h>

using namespace std;

//--------------------------------------------------------------------------
// Particle Class
//
class Particle{

        public:
        Particle();
        // FIXME : Create an additional constructor that takes 4 arguments --> the 4-momentum
        Particle (double, double, double, double);
        double   pt, eta, phi, E, m, p[4];
        void     p4(double, double, double, double);
        void     print();
        void     setMass(double);
        double   sintheta();
};

//--------------------------------------------------------------------------

//*************************************************************************
//                                                                        *
//    MEMBERS functions of the Particle Class                             *
//                                                                        *
//*************************************************************************

//
//*** Default constructor ------------------------------------------------
//
Particle::Particle(){
        pt = eta = phi = E = m = 0.0;
        p[0] = p[1] = p[2] = p[3] = 0.0;
}

//*** Additional constructor ---------------------------------------------------
Particle::Particle(double p0, double p1, double p2, double p3){
        //FIXME
        TLorentzVector part;
        part.SetXYZT(p1,p2,p3,p0);

        this->p[0] = part[0];
        this->p[1] = part[1];
        this->p[2] = part[2];
        this->p[3] = part[3];
        this->pt = part.Pt();
        this->eta = part.Eta();
        this->phi = part.Phi();
        this->E = part.E();
        this->m = part.M();
}

//
//*** Members  -----------------------------------------------------
//
double Particle::sintheta(){
        //FIXME
        TLorentzVector particle;
        particle.SetXYZT(this->p[1],this->p[2],this->p[3],this->p[0]);
        return sin(particle.Theta());
}

void Particle::p4(double pT, double eta, double phi, double energy){
        // FIXME
        TLorentzVector particle;
        particle.SetPtEtaPhiE(pT, eta, phi, energy);
        this->p[0] = particle[0];
        this->p[1] = particle[1];
```

```
   84:         this->p[2] = particle[2];
   85:         this->p[3] = particle[3];
   86: }
   87:
   88: void Particle::setMass(double mass)
   89: {
   90:         // FIXME
   91:         this->m = mass;
   92: }
   93:
   94: //
   95: //*** Prints 4-vector ------------------------------------------------------
   96: //
   97: void Particle::print(){
   98:         std::cout << "p4 = (" << p[0] <<",\t" << p[1] <<",\t"<< p[2] <<",\t"<< p[3] << ")" << "  sin(theta)
=" <<  sintheta() << std::endl;
   99: }
  100:
  101:
  102: class Lepton : public Particle {
  103:         using Particle::Particle;
  104:         public:
  105:         signed int      Q;
  106:         void set_charge(signed int charge){
  107:                 this->Q = charge;
  108:         };
  109: };
  110:
  111: class Jet : public Particle {
  112:         using Particle::Particle;
  113:         public:
  114:         int     f;
  115:         void set_flavor(int flavor){
  116:                 this->f = flavor;
  117:         };
  118: };
  119:
  120: int main(int argc, char ** argv) {
  121:
  122:         /* ************* */
  123:         /* Input Tree    */
  124:         /* ************* */
  125:
  126:         TFile *f  = new TFile(argv[1],"READ");
  127:         TTree *t1 = (TTree*)(f->Get("t1"));
  128:
  129:         // Read the variables from the ROOT tree branches
  130:         t1->SetBranchAddress("lepPt",&lepPt);
  131:         t1->SetBranchAddress("lepEta",&lepEta);
  132:         t1->SetBranchAddress("lepPhi",&lepPhi);
  133:         t1->SetBranchAddress("lepE",&lepE);
  134:         t1->SetBranchAddress("lepQ",&lepQ);
  135:
  136:         t1->SetBranchAddress("njets",&njets); // not defined in t1.h
  137:         t1->SetBranchAddress("jetPt",&jetPt);
  138:         t1->SetBranchAddress("jetEta",&jetEta);
  139:         t1->SetBranchAddress("jetPhi",&jetPhi);
  140:         t1->SetBranchAddress("jetE", &jetE);
  141:         t1->SetBranchAddress("jetHadronFlavour",&jetHadronFlavour);
  142:
  143:         // Total number of events in ROOT tree
  144:         Long64_t nentries = t1->GetEntries();
  145:
  146:         for (Long64_t jentry=0; jentry<3;jentry++)
  147:         {
  148:                 t1->GetEntry(jentry);
  149:                 std::cout<<"\n\n Event "<< jentry <<std::endl;
  150:
  151:                 //FIX ME
  152:                 // cout << njets << ", " << sizeof(jetE)<< ", " << sizeof(lepE)<< ", " << endl;
  153:                 cout << "\n Jet particles:" << endl;
  154:                 for (Long_t part=0; part<sizeof(jetE); part++)
  155:                 {
  156:                         Jet jet_object;
  157:                         jet_object.p4(jetPt[part], jetEta[part], jetPhi[part], jetE[part]);
  158:                         jet_object.set_flavor(jetHadronFlavour[part]);
  159:                         jet_object.print();
  160:                 }
  161:
  162:                 cout << "\n Lepton particles:" << endl;
  163:                 for (Long_t part=0; part<sizeof(jetE); part++)
  164:                 {
  165:                         Lepton lepton_object;
```

```
166:                         lepton_object.p4(lepPt[part], lepEta[part], lepPhi[part], lepE[part]);
167:                         lepton_object.set_charge(lepQ[part]);
168:                         lepton_object.print();
169:                 }
170:
171:         } // Loop over all events
172:
173:         return 0;
174: }
```