```cpp
#include <stdio.h>
#include <iostream>
#include <stddef.h>
#include <string>
#include <cstdlib>
#include <cstdarg>
#include <cmath>
#include <fstream>
#include <vector>
#include <cstdio>
#include <fstream>
#include <stdlib.h>
#include <sstream>

using std::cout;
using std::endl;
using std::ios;

double Evaluate_dUdy_Plus(double lmix_plus, double y_pl); // function prototype

int main(){
  cout<<"RK4 integration scheme applied to a first order non-linear ODE"<<endl;

  double lmix_pl, y_pl;

  int Karman = 0;      // Set to 0 to turn off, 1 to activate
  int VanDriest = 1;
  double A0_pl = 26;
  double K = 0.41;
  int counter = 0;
  int size = 100000;

  double delta_y_pl;
  double y_pl_max = 500;
  std::vector <double> U_pl_profile;
  std::vector <double> Y_pl_profile;

  double dU_dy0, dU_dy1, dU_dy2, dU_dy3, dU_dy, U_n1 = 0.0, U_n2 = 0;

  int filewrite = 1;  // set to 1 to output file

  y_pl = 0.0;

  while (y_pl <= y_pl_max){
    if (Karman == 1){
      lmix_pl = K * y_pl;
    }
    else if (VanDriest == 1)
    {
      lmix_pl = K*y_pl*(1 - exp(-y_pl/A0_pl) );
    }
    else{
      cout<<"please specify the needed mixing length model"<<endl;
      exit(1);
    }

    if (y_pl <=0.1){
      delta_y_pl = 1e-1;
    }
    else if (y_pl > 0.1){
      delta_y_pl = 5e-1; // 1;
    }

    /* ==========  RK4 Scheme   ================
     *     k_1  =       hf(x_n,y_n)
```

```cpp
     *     k_2  =       hf(x_n+1/2h,y_n+1/2k_1)
     *     k_3  =       hf(x_n+1/2h,y_n+1/2k_2)
     *     k_4  =       hf(x_n+h,y_n+k_3)
     *     y_(n+1)      =       y_n+1/6k_1+1/3k_2+1/3k_3+1/6k_4+O(h^5)
     *   ======================================================================= */

    dU_dy0 = Evaluate_dUdy_Plus(lmix_pl, y_pl);
    dU_dy1 = Evaluate_dUdy_Plus(lmix_pl + 0.5*delta_y_pl*dU_dy0, y_pl + 0.5*delta_y_pl);
    dU_dy2 = Evaluate_dUdy_Plus(lmix_pl + 0.5*delta_y_pl*dU_dy1, y_pl + 0.5*delta_y_pl);
    dU_dy3 = Evaluate_dUdy_Plus(lmix_pl + 0.5*delta_y_pl*dU_dy2, y_pl + delta_y_pl);

    if (counter > 0){
      U_n2 = U_n1 + (delta_y_pl/6)*(dU_dy0 + 2*dU_dy1 + 2*dU_dy2 + dU_dy3);
    }
//     cout<<"U+ is: "<<U_n2<<"  y+ is: "<<y_pl<<" lmix+ is: "<<lmix_pl<<endl;

    U_pl_profile.push_back(U_n2);
    Y_pl_profile.push_back(y_pl);

    if (fabs(y_pl - y_pl_max) <= 5e-1){
      break;
    }
    y_pl = y_pl + delta_y_pl;
    counter = counter + 1;
    U_n1 = U_n2;
  }
  //=============== now write results to file (Tecplot!)

  if (filewrite == 1){
    std::stringstream stream1;
    std::stringstream stream3;
    std::stringstream stream4;
    if (Karman == 1){
      stream1 << "U_plus_Karman_Mixing_Length_Model.dat";
    }
    else if (VanDriest == 1){
      stream1 << "U_plus_VanDriest_Mixing_Length_Model.dat";
    }
    stream3 <<"i="<<counter + 1;
    stream4<<"title = "<<"'"<<stream1.str()<<"'";
    std::string var1 = stream3.str();
    std::string var2 = stream4.str();
    std::string fileName1 = stream1.str();
    FILE* fout = fopen(fileName1.c_str(), "w");
    fprintf(fout, "%s", var2.c_str() ); fprintf(fout, "\n");
    fprintf(fout, "%s", "variables = 'y+', 'U+' "); fprintf(fout, "\n");
    fprintf(fout, "%s %s %s", "zone",var1.c_str(),"f=point"); fprintf(fout, "\n");  //
;
    for (int j = 0; j<=counter; j++){
      fprintf(fout, "%e\t %e\t", Y_pl_profile[j], U_pl_profile[j]);
      fprintf(fout, "\n");
    }
    fclose(fout);
    if (Karman == 1){
      std::cout<<"Von Karman Profile successfully written"<<endl;
    }
    else{
      std::cout<<"Van Driest Profile successfully written"<<endl;
    }

  }

  return 0;

}
```

```
double Evaluate_dUdy_Plus(double lmix_plus, double y_pl){

  double dU_dy = 0.0;

  if ( y_pl >=0.0 && y_pl <= 0.1){
    dU_dy = 1 - (lmix_plus * lmix_plus) + 2*(lmix_plus*lmix_plus*lmix_plus*lmix_plus );
  }
  else if (y_pl > 0.1 && y_pl <500.0 ){  // usual
    dU_dy =  (std::sqrt( 4 *lmix_plus*lmix_plus  + 1 ) - 1 ) / (2 * (lmix_plus*lmix_plus) ) ;
  }

  else{
    cout<<"y+ is out of range"<<endl;
  }

  return dU_dy;

}
```