```c
// PROGRAMME D'OPTIMISATION : trouver ms, phis, alpha optimales

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//Fonction pour calculer phi
double phi(double alpha, double ms, double phis, double m){
   double bla=0.4*(ms-m)*(alpha+1);
   double blabla=(0.4*(ms-m));
   return(0.4*log(10)*phis*pow(10,bla)*exp(-pow(10,blabla)));}

//La fonction d'optimisation khi2
double khi2(double phi, int N){
  double bla=(N-phi)/((sqrt(N))); //le dénominateur tient compte des poids
  return(pow(bla,2));}

main () {
  double k=0, pas=0.001, k2=1000000;
  double  p;
  int j, o;
  int i=0, n, N[52];
  double alpha, ms, phis, m, M[52];

  FILE*histo; // nos données sont dans le fichier histo2
  histo=fopen("/home/ebellomi/projet30/Programmes/histo2.ascii", "r");
  FILE*fic; // et on veut les écrire dans le fichier khi2
  fic=fopen("/home/ebellomi/projet30/Programmes/khi2.ascii", "w");

  for(i=0;i<52;i++){
    fscanf(histo, " %lf %d \n",&M[i],&N[i]);
    } // remplissage du tableau

  for(ms=19.0; ms>16.95; ms-=0.1){
    for(o=-10;o<=10;o++){
     j=(ms-16.95)/0.1;
     phis=N[j]+o;
        for(alpha=-3.0; alpha<=0.0; alpha+=pas){ //on cherche la pente optimale
                k=0;
                fclose(histo);
                histo=fopen("/home/ebellomi/projet30/Programmes/histo2.ascii", "r");

                while (fscanf(histo, " %lf %d \n",&m,&n) != EOF){
                      if(n!=1){
                        p=phi(alpha, ms, phis, m);
                        k+=khi2(p,n);}
                    }// while

                if(k<k2){
                //il mémorise les nouveaux paramètres seulement s'ils optimisent mieux la fonction
                        fprintf(fic, " %lf %lf %lf %lf \n", k, ms, phis, alpha);
                        printf(" %lf %lf %lf %lf \n", k, ms, phis, alpha);
                        k2=k;
                        } // if

        } /* for alpha*/ } /*for o phis*/  } // for ms

  fclose(histo);
  fclose(fic);
} // main


//LIGNES DE COMMANDES UTILES

awk '{if (NR>1 && $1<4) print $0}' //il imprime tout les colonnes à partir de la deuxième ligne et si
l'élément de la première colonne est un nombre plus petit de 4

sort file.ascii > filesort.ascii // il ordonne les valeurs de la première colonnes du plus petit au
plus grand
```