```
   Help
#include "optype.h"
#include "var.h"
#include "method.h"
#include "test.h"
#include "timeinfo.h"
#include "error_msg.h"
#include "tools.h"
#include "ftools.h"
#ifndef SEEK_SET
#define SEEK_SET 0
#endif
#ifndef CLOCKS_PER_SEC
#include <unistd.h>
#define CLOCKS_PER_SEC _SC_CLK_TCK
#endif
extern char **error_msg;
extern char premiasrcdir[256];
extern char premiapersodir[256];
extern char *path_sep;
extern char PREMIA_OUT[256];
extern char GNUPLOT_DAT[256];
extern char TITLES_TEX[256];
extern char GNUPLOT_SCREEN_PLT[256];
extern char GNUPLOT_FILE_PLT[256];
extern char GNU_TEX[256];
extern char PREMIA_LOG[256];
extern char SESSION_LOG[256];

void ReadInputFile(char *InputFile, char FileRed[
    MAX_LINE][MAX_CHAR_LINE])
{
  FILE *fic;
  char line[MAX_CHAR_LINE];
  int i;


  printf("Reading File %s...{n",InputFile);


  if((fic = fopen(InputFile,"r")) == NULL)
```

```
    {
      printf("Unable to open Input File %s{n",
    InputFile);
      exit(1);
    }
  i=0;

  printf("%d %d {n",MAX_LINE, MAX_CHAR_LINE);


  while( i < MAX_LINE){
    if(fgets(line,MAX_CHAR_LINE,fic) == NULL){

      if( fgetc(fic) == EOF)
        i=MAX_LINE+5;
      else{
        printf("Unable to read Input File %s{
    n",InputFile);
        exit(1);
      }
    }

    if(line[0] == '#'){
      strcpy(FileRed[0],line);
    }
    i++;
  }
  if( i == MAX_LINE+6)
    printf("The File %s is too long. The read
    ing process stops at line 100. May be lost of data
    ....{n",InputFile);


}



char FChooseAction(char **InputFile)
{
  char msg='e';
```

```
  return msg;
}

int FMoreAction(int *count)
{
  char msg='e';
  if (*count==(MAX_METHODS-1))
    {
      Fprintf(TOSCREEN,"{n Max Number of Methd
    s Reached!{n");
      msg='n';
    } else
      {
        Fprintf(TOSCREEN,"{nMore Methods (ok:
     Return, no: n)?:{t");


        while ((msg!='{n')&&(msg!='n'))
          scanf("%c",&msg);
        fflush(stdin);
      }
  if(msg=='n')
    return WRONG;
  else{
    (*count)++;
    return OK;
  }
}

int FSelectModel(int user,Planning *pt_plan,Model
     **listmodel,Model **mod)
{
  int choice=0, i=0;
  char fhelp_name[MAX_PATH_LEN]="";
  char msg,answer;
```

```
  if ((strlen(premiasrcdir)+strlen(path_sep)+str
    len("mod_doc.pdf"))>=MAX_PATH_LEN)
    {
      Fprintf(TOSCREEN,"%s{n",error_msg[PATH_
    TOO_LONG]);
      exit(WRONG);
    }

  strcpy(fhelp_name,"..");
  strcat(fhelp_name,path_sep);
  strcat(fhelp_name,"Src");
  strcat(fhelp_name,path_sep);
  strcat(fhelp_name,"mod_doc.pdf");

  Fprintf(TOSCREEN,"{n_________________________MOD
    EL CHOICE:{n{n");

  while (listmodel[i]!=NULL)
    {
      Fprintf(TOSCREEN,"%s:{t%d{n",listmodel[i]->
    ID,i+1);
      i=i+1;
    }

  Fprintf(TOSCREEN,"{nChoice (h or any letter fo
    r help)?:{t");
  do
    {
      msg = (char)tolower(fgetc(stdin));
      answer = msg;
      while( msg != '{n' && msg != EOF)
        msg = (char)fgetc(stdin);

      if (isalpha(answer) != 0)
        {
          choice=0;
#ifdef _WIN32
          _spawnlp(_P_WAIT,"AcroRd32.exe","_
    spawnlp",fhelp_name,NULL);
#endif
#ifndef _WIN32
```

```
          _spawnlp(1,"acroread","_spawnlp",
    fhelp_name,NULL);
#endif
          Fprintf(TOSCREEN,"{nNew value?:{t"
    );

        }
      else
        if(isdigit(answer) != 0)
          {
            choice = atoi(&answer);

            if ((choice<=0)||(choice>i))
              {
                Fprintf(TOSCREEN,"{nBad
    Choice: should range between 1 and %d ! New value?
    :{t",i);
              }
          }

    } while ((choice<=0)||(choice>i));
  Fprintf(TOSCREEN,"{n");

  if ((0<choice)&&(choice<=i))
    {
      *mod=listmodel[choice-1];
      return ((*mod)->Get)(user,pt_plan,*mod);
    }

  Fprintf(TOSCREEN,"Bad Choice!{n");

  return WRONG;
}


int FSelectOption(int user,Planning *pt_plan,Fami
    ly **L_listopt,Model* pt_model,Pricing **pricing,
    Option **opt)
{
  int i,j,choice,k;
  Family* list;
```

```
char family_name[MAX_CHAR_X3]="",dummy[MAX_CHA
  R_X3]="";
char msg,answer[3];

do{
  Fprintf(TOSCREEN,"{n_____________________
  OPTION CHOICE:{n{n");

  i=0;
  while (L_listopt[i]!=NULL)
    {
      if (MatchingPricing(user,pt_model,*(
  L_listopt[i])[0],pricing)==0)
        {
          list=L_listopt[i];
          if ((strlen(premiasrcdir)+strle
  n(path_sep))>=MAX_CHAR_X3)
            {
              Fprintf(TOSCREEN,"%s{n",
  error_msg[PATH_TOO_LONG]);
              exit(WRONG);
            }
          strcpy(family_name,"..");
          strcat(family_name,path_sep);
          strcat(family_name,"Src");
          strcat(family_name,path_sep);
          if ((strlen("Opt")+2*strlen(pat
  h_sep)+2*strlen((*list)[0]->ID)
               +strlen("_doc.pdf"))>=
  MAX_CHAR_X3)
            {
              Fprintf(TOSCREEN,"%s{n",
  error_msg[PATH_TOO_LONG]);
              exit(WRONG);
            }

          strcpy(dummy,"opt");
          strcat(dummy,path_sep);
          strcat(dummy,(*list)[0]->ID);
          strcat(dummy,path_sep);
          strcat(dummy,(*list)[0]->ID);
```

```
        strcat(dummy,"_doc.pdf");
        for(k=0;k<(int)strlen(dummy);k+
+)
          dummy[k] = (char)tolower(dum
my[k]);
        if ((strlen(family_name)+strle
n(dummy)>=MAX_CHAR_X3))
          {
            Fprintf(TOSCREEN,"%s{n",
error_msg[PATH_TOO_LONG]);
            exit(WRONG);
          }

        strcat(family_name,dummy);
        Fprintf(TOSCREEN,"{nFamily{t%s{
n",(*list)[0]->ID);
        j=0;
        while ((*list)[j]!=NULL)
          {
            Fprintf(TOSCREEN,"%s:{t%
d{n",(*list)[j]->Name,j+1);
            j=j+1;
          }

        Fprintf(TOSCREEN,"{nChoice (0
for NextFamily, h for help)?:{t");
        do
          {
            k=0;
            msg=(char)tolower(fgetc(
stdin));
            answer[k++] = msg;
            while( (msg != '{n') && (
msg != EOF)){
              msg = (char)fgetc(stdi
n);
              if(k<=2)
                answer[k++]=msg;
            }
            answer[k--]='{0';
            if (isdigit(answer[0]) ==
```

```
     0)
                  {
                    choice=j+1;
                    if(answer[0] == 'h'
    ){
#ifdef _WIN32
                      _spawnlp(_P_WAI
    T,"AcroRd32.exe","_spawnlp",family_name,NULL);
#endif
#ifndef _WIN32
                      _spawnlp(1,"acro
    read","_spawnlp",family_name,NULL);
#endif
                    }
                    Fprintf(TOSCREEN,"{
    nNew value?:{t");

                  } else {
                    choice = atoi(answ
    er);
                    if ((choice<0)||(
    choice>j))
                      {

                        Fprintf(TOSCR
    EEN,"{nBad Choice: should range between 1 and %d
    ! New value?:{t",j);
                      }
                  }
             } while ((choice<0)||(choic
    e>j));

            Fprintf(TOSCREEN,"{n");

            if ((0<choice)&&(choice<=j))
              {
                *opt=(*list)[choice-1];
                return ((*opt)->Get)(use
    r,pt_plan,*opt);;
              }
```

```
            Fprintf(TOSCREEN,"{n");
          }
        i=i+1;/*choice=0*/
      }

    Fprintf(TOSCREEN,"No more families!{n");
  } while(1);

  return WRONG;
}


int FSelectPricing(int user,Model *pt_model,Opti
    on *pt_option,Pricing **pricing,Pricing **result)
{
  int i=-1;
  char dummy[MAX_CHAR_X3];

  if ((strlen(pt_model->ID)+1+strlen(pt_option->
    ID))>=MAX_CHAR_X3)
    {
      Fprintf(TOSCREEN,"%s{n",error_msg[PATH_
    TOO_LONG]);
      exit(WRONG);
    }

  strcpy(dummy,pt_model->ID);
  strcat(dummy,"_");
  strcat(dummy,pt_option->ID);

  do
    {
      i=i+1;
    }
  while ((strcmp(dummy,pricing[i]->ID)!=0) && (pr
    icing[i+1]!=NULL));

  if (strcmp(dummy,pricing[i]->ID)==0)
    {
      *result=pricing[i];
```

```
      return ((*result)->CheckMixing)(pt_option,
    pt_model) ;
    }
  Fprintf(TOSCREEN,"No choice available!{n");

  return WRONG;
}

int FSelectMethod(int user,Planning *pt_plan,Pric
    ing *pt_pricing, Option *opt,Model *mod,Pricing
    Method **met)
{
  int i,isub,choice,sublist[MAX_MET],k;
  char msg,answer[3];
  PricingMethod** list;
  PricingMethod* dummy;

  Fprintf(TOSCREEN,"{n_________________________
    METHOD CHOICE:{n{n");

  list=pt_pricing->Methods;
  i=0;isub=0;

  dummy=*list;

  while (dummy !=NULL)

    {
      if ( (dummy->CheckOpt)(opt,mod)==OK)

        {
          Fprintf(TOSCREEN,"%s:{t%d{n",(pt_
    pricing->Methods[i])->Name,isub+1);
          sublist[isub]=i;
          isub=isub+1;
        }

      i=i+1;
      list++;dummy=*list;
    }
```

```
list=pt_pricing->Methods;
if (isub==0)
  {
    Fprintf(TOSCREEN,"No methods available!{n")
  ;
  }
else
  {
    Fprintf(TOSCREEN,"{nChoice?:{t");
    do
      {
        k=0;
        msg = (char)tolower(fgetc(stdin));
        answer[k++] = msg;
        while( (msg != '{n') && (msg != EO
  F)){
          msg = (char)fgetc(stdin);
          if(k<=2)
            answer[k++]=msg;
        }
        answer[k--]='{0';
        if (isdigit(answer[0]) == 0) {
          Fprintf(TOSCREEN,"{nNew value?:
  {t");
          choice=0;
        }
        else{
          choice = atoi(answer);
          if ((choice<=0)||(choice>isub))
            {
              Fprintf(TOSCREEN,"{nBad
  Choice: should range between 1 and %d ! New value?
  :{t",isub);
            }
        }
    }    while ((choice<=0)||(choice>is
  ub));
    Fprintf(TOSCREEN,"{n");

    if ((0<choice)&&(choice<=isub))
      {
```

```
          *met=*(list+sublist[choice-1]);
          return GetMethod(user,pt_plan,pt_
    pricing,*met);
        }
      else
        Fprintf(TOSCREEN,"Bad choice!{n");
    }
  return WRONG;

}
```

References