```
   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"
#include "premia_obj.h"
#ifndef SEEK_SET
#define SEEK_SET 0
#endif
#ifndef CLOCKS_PER_SEC
#include <unistd.h>
#include "error_msg.h"
#define CLOCKS_PER_SEC _SC_CLK_TCK
#endif



extern char premiasrcdir[MAX_PATH_LEN];
extern char premiamandir[MAX_PATH_LEN];

extern char PREMIA_OUT[MAX_PATH_LEN];
extern char GNUPLOT_DAT[MAX_PATH_LEN];
extern char TITLES_TEX[MAX_PATH_LEN];
extern char GNUPLOT_SCREEN_PLT[MAX_PATH_LEN];
extern char GNUPLOT_FILE_PLT[MAX_PATH_LEN];
extern char GNU_TEX[MAX_PATH_LEN];
extern char PREMIA_LOG[MAX_PATH_LEN];
extern char SESSION_LOG[MAX_PATH_LEN];

int StrCasecmp(const char *chaine1,const char *chaine2)
{
  int i,ma,test;

  if (strlen(chaine1)!=strlen(chaine2))
    return 1;
  else{
    ma=strlen(chaine1);
    i=0;test=0;
    while(i<ma){
      if(tolower(*chaine1) == tolower(*chaine2))
        {
          chaine1++;
          chaine2++;
          test++;
          i++;
        }else break;
    }
    if((i==ma) && (test==ma))
      return 0;
    else
      return -1;
  }
}
void ReadInputFile(char *InputFile, char FileRed[MAX_LINE][
    MAX_CHAR_LINE])
{
  FILE *fic;
  char c;
  int i,j;

  for(i=0;i<MAX_LINE;i++)
    for(j=0;j<MAX_CHAR_LINE;j++)
      FileRed[i][j]='{0';
  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;j=0;


  while ( (i < MAX_LINE) && ((c = fgetc(fic)) != EOF) )
    {
      switch(c)
        {
```

```
        case '#': /* Ignore commented lines */
          while((c=fgetc(fic)) != '{n');
          break;
        case '{r': /* windows end of line character */
          break;
        case '{n': /* we start a new line */
          if ( j>0 ) { i++; j=0; }
          break;
        case ' ': /* Keep blanks only if we are at the beg
    inning of line */
          if ( j>0 ) { FileRed[i][j] = c; j++; }
          break;
        default:
          FileRed[i][j] = c; j++;
          break;
        }
    }
  fclose(fic);

  if( i == MAX_LINE)
    printf("The File %s is too long. The reading process
    stops at line %d. May be lost of data....{n", InputFile, MAX
    _LINE);

}

char FChooseProduct(char InputFile[MAX_LINE][MAX_CHAR_LINE]
    )
{
  char msg='e';
  int i,i0,j,j0;
  printf("{nReading Product type.....{n");
  i0=-1;j0=-1;
  for ( i=0 ; (i<MAX_LINE)  && (i0<0) ; i++ )
    {
      j=0;
      while (j< (signed)strlen(InputFile[i])-3)
        {
          if( ((InputFile[i][j] == 'P') ||  (InputFile[i][
    j] == 'p'))
              && ((InputFile[i][j+1] == 'r') ||  (InputFil
    e[i][j+1] == 'R'))
              && ((InputFile[i][j+2] == 'o') ||  (InputFil
    e[i][j+2] == 'O')) )
            {
              i0 = i;
              j0 = j+3;
              while (InputFile[i][j0] == ' ') j0++;
              break;
            }
          j++;
        }
    }
  if ( i0 > -1 )
    {
      PremiaAsset *asset;
      msg = InputFile[i0][j0];
      for ( asset = premia_assets ; asset->name != NULL ;
    asset++ )
        {
          if ( msg == asset->label )
            {
              printf("Asset %s found{n", asset->name);
              return msg;
            }
        }
    }
  printf("Asset is missing, default is equity{n");
  return 'e';
}


char FChooseAction(char InputFile[MAX_LINE][MAX_CHAR_LINE])
{
  char msg='p';
  int i,i0,j,j0;
  printf("{nReading Action type.....{n");
  i0=-1;j0=-1;
  for(i=0;((i<MAX_LINE)  && (i0<0));i++){
    j=0;
    while (j< (signed)strlen(InputFile[i])-3) {
      if( ((InputFile[i][j] == 'A') ||  (InputFile[i][j] ==
     'a'))
          && ((InputFile[i][j+1] == 'C') ||  (InputFile[i][
    j+1] == 'c'))
          && ((InputFile[i][j+2] == 'T') ||  (InputFile[i][
    j+2] == 't'))){
        i0 = i;
        j0 = j+3;
        while (InputFile[i][j0] == ' ') j0++;
      }
      j++;
    }
  }
  if ((i0<0) || ((InputFile[i0][j0]!='p')&& (InputFile[i0][
    j0]!='t'))){
    printf("Action is missing, default is pricing{n");
    msg='p';
  }else{
    msg = InputFile[i0][j0];
    if (msg == 'p')
      printf("Action found: Pricing{n{n");
    else
      printf("Action found: Test{n{n");
  }

  return msg;
}

int FMoreAction(char InputFile[MAX_LINE][MAX_CHAR_LINE],
    int *count)
{
  //char msg='p';
  int i0=0,i,j;
  //int listline[MAX_LINE];
  if (*count==(MAX_METHODS-1))
    {
      Fprintf(TOSCREEN,"{n Max Number of Methds Reached!{n"
    );
      //msg='n';
    }else{
      i0=0;
      for(i=0;i<MAX_LINE;i++){
        if(strlen(InputFile[i])>1){
          j=0;
          while(isalpha(InputFile[i][j])==0)
            j++;
          if(((InputFile[i][j]=='M') || (InputFile[i][j]=='
    m')) && ((InputFile[i][j+1]=='A')

             ||(InputFile[i][j+1]=='a'))){
            //listline[i0]=i;
            i0++;
          }
        }
      }


    }
  if(i0==(*count))
    return WRONG;
  else{
    (*count)++;
    return OK;
  }
}

int FSelectModel(char InputFile[MAX_LINE][MAX_CHAR_LINE],
    int user,Planning *pt_plan,Model **listmodel,Model **mod)
{
  int choice=0, i=0,mo, aux;
  /* char fhelp_name[MAX_PATH_LEN]="";*/
  int ims,j,k;
  char line[MAX_CHAR_LINE];
  char **Inp=NULL;
  Fprintf(TOSCREEN,"{n_______________________MODEL CHOICE:{
    n{n");

  /* Find  a model in the file */
  /* Find the designation of the model*/
  ims=-1;
  for(i=0;((i<MAX_LINE) && (ims<0));i++){
    mo=0;
    while (listmodel[mo]!=NULL)
```

```
      {
        j=0;
        while(j<=(signed)(strlen(InputFile[i])-strlen(listm
    odel[mo]->ID)))
          {
            for(k=j;k<j+(signed)strlen(listmodel[mo]->ID);
    k++)
              line[k-j] = InputFile[i][k];
            line[j+(signed)strlen(listmodel[mo]->ID)]='{0';
            if (StrCasecmp(listmodel[mo]->ID,line) == 0){
              ims = i;
              choice = mo;
            }
            j++;
          }
        mo++;

      }
  }
  if(ims<0){
    printf("{nNot able to find a model: default is Black
    Scholes 1d{n");
    return PREMIA_NONE;
  }else{
    printf("{nA model has been found: %s{n{n",listmodel[cho
    ice]->ID);
    *mod=listmodel[choice];
    Inp = malloc(sizeof(char *)*MAX_LINE);
    for(i=0;i<MAX_LINE;i++){
      Inp[i]= malloc(sizeof(char)*(strlen(InputFile[i])+1))
    ;
      for(j=0;j<(signed)strlen(InputFile[i]);j++)
        Inp[i][j]=InputFile[i][j];
      Inp[i][strlen(InputFile[i])]='{0';
    }
    aux = ((*mod)->FGet)(Inp,user,pt_plan,*mod);
    for(i=0;i<MAX_LINE;i++) free(Inp[i]);
    free(Inp); Inp = NULL;
    return aux;
  }
}
```

```
int FSelectOption(char InputFile[MAX_LINE][MAX_CHAR_LINE],
    int user,Planning *pt_plan,Family **L_listopt,Model* pt_
    model,Pricing **pricing,Option **opt)
{
  int i,j,choice,k,ims,op,mo,cat, aux;
  Family* list;
  /* char family_name[MAX_CHAR_X3]="",dummy[MAX_CHAR_X3]=""
    ; */
  char line[MAX_CHAR_LINE];
  char **Inp = NULL;
  /* avoid warning */
  cat = choice =0;

  Fprintf(TOSCREEN,"{n_______________________OPTION CHOICE:
    {n{n");
  ims=-1;
  for(i=0;((i<MAX_LINE) && (ims<0));i++){

    op=0;
    while (L_listopt[op]!=NULL)
      {
        if (MatchingPricing(pt_model,*(L_listopt[op])[0],
    pricing)==0)
          {
            list=L_listopt[op];
            mo=0;
            while ((*list)[mo]!=NULL)
              {
                j=0;
                while(j<=(signed)(strlen(InputFile[i])-strl
    en((*list)[mo]->Name)))
                  {
                    if( (j==0 || (j>0 && InputFile[i][j-1]=
    =' ')) &&
                        ( InputFile[i][j+(signed)strlen((*
    list)[mo]->Name)] == ' ' ||
                          InputFile[i][j+(signed)strlen((*
    list)[mo]->Name)] == '{0')
                      ){
                      for(k=j;k<j+(signed)strlen((*list)[
    mo]->Name);k++)
                        line[k-j] = InputFile[i][k];
                      line[j+(signed)strlen((*list)[mo]->Na
    me)]='{0';

                      if (StrCasecmp((*list)[mo]->Name,line
    ) == 0){
                        ims = i;
                        choice = mo;
                        cat=op;
                      }
                    }
                    j++;
                  }
                mo++;
              }
          }

        op++;
      }
  }

  if(ims<0){
    printf("{nNot able to find an option: default is     CallEuro 1d{n");
    return PREMIA_NONE;
  }else{
    printf("{nAn Option has been found: %s{n{n",(*L_listop
    t[cat])[choice]->Name);
    *opt=(*L_listopt[cat])[choice];
    Inp = malloc(sizeof(char *)*MAX_LINE);
    for(i=0;i<MAX_LINE;i++){
      Inp[i]= malloc(sizeof(char)*(strlen(InputFile[i])+1))
    ;
      for(j=0;j<(signed)strlen(InputFile[i]);j++)
        Inp[i][j]=InputFile[i][j];
      Inp[i][strlen(InputFile[i])]='{0';
    }
    aux = ((*opt)->FGet)(Inp,user,pt_plan,*opt, pt_model);
    for(i=0;i<MAX_LINE;i++) free(Inp[i]);
    free(Inp); Inp = NULL;
    return aux;
  }
}


int FSelectPricing(char InputFile[MAX_LINE][MAX_CHAR_LINE],
    int user,Model *pt_model,Option *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) && (pricing[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 PREMIA_NONE;
}

int FSelectMethod(char InputFile[MAX_LINE][MAX_CHAR_LINE],
    int user,Planning *pt_plan,Pricing *pt_pricing, Option *opt,
    Model *mod,PricingMethod **met)
{
  int i,isub,ii,j,choice,sublist[MAX_MET],k,is,ie, aux;
  char line[MAX_CHAR_LINE];
  int listline[MAX_MET],choiceline[MAX_MET];
  char **Inp;
  PricingMethod** list;
  PricingMethod* dummy;

  Fprintf(TOSCREEN,"{n_______________________METHOD CHOICE:
    {n{n");

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

  dummy=*list;
  choice=0;
  while (dummy !=NULL)

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

        {

          for(ii=0;ii<MAX_LINE;ii++){
            j=0;
            while(j<=(signed)(strlen(InputFile[ii])-strlen(
    (pt_pricing->Methods[i])->Name)))
              {
                if( (j==0 || (j>0 && InputFile[i][j-1]==' '
    )) &&
                    ( InputFile[i][j+strlen((pt_pricing->
    Methods[i])->Name)] == ' ' ||
                      InputFile[i][j+strlen((pt_pricing->
    Methods[i])->Name)] == '{0')
                  ){
                  for(k=j;k<j+(signed)strlen((pt_pricing->
    Methods[i])->Name);k++)
                    line[k-j] = InputFile[ii][k];
                  line[j+(signed)strlen((pt_pricing->
    Methods[i])->Name)]='{0';
```

```
                if (StrCasecmp((pt_pricing->Methods[i])->
  Name,line) == 0){
                  listline[choice]=isub;
                  choiceline[choice]=ii;
                  choice++;
                }
              }
              j++;
            }
        }
        sublist[isub]=i;
        isub=isub+1;
      }

    i=i+1;
    list++;dummy=*list;
  }
/* Tri des methodes */
for(j=1;j<choice;j++){
  ii = choiceline[j];
  k=listline[j];
  i=j-1;
  while(i>=0 && choiceline[i]>ii){
    choiceline[i+1]=choiceline[i];
    listline[i+1]=listline[i];
    i--;
  }
  choiceline[i+1]=ii;
  listline[i+1]=k;
}
/* On envoie que la partie du fichier necessaire pour ev
  iter les confusions
   de parametres de methodes */
list=pt_pricing->Methods;
if (isub==0){
  Fprintf(TOSCREEN,"No methods available!{n");
}else{
  if(choice <0){
    Fprintf(TOSCREEN,"No methods found, exiting....{n");
  }else{
    Fprintf(TOSCREEN,"{n");
```

```
      if(choice <= pt_plan->NumberOfMethods){
        Fprintf(TOSCREEN,"Problem with the name of a
    Method{n");
      }else{
        if(choice -1 == pt_plan->NumberOfMethods){
          is = choiceline[pt_plan->NumberOfMethods];
          ie =MAX_LINE;
        }else{
          is = choiceline[pt_plan->NumberOfMethods];
          ie = choiceline[pt_plan->NumberOfMethods+1];
        }
        Inp = malloc(sizeof(char *)*MAX_LINE);
        for(i=0;i<MAX_LINE;i++)
          Inp[i]= malloc(sizeof(char)*(MAX_CHAR_LINE));
        for(i=is;i<ie;i++){
          for(j=0;j<(signed)strlen(InputFile[i]);j++)
            Inp[i-is][j]=InputFile[i][j];
          Inp[i-is][strlen(InputFile[i])]='{0';
        }
        *met=*(list+sublist[listline[pt_plan->NumberOfMetho
    ds]]);
        aux =  FGetMethod(Inp,user,pt_plan,pt_pricing,*met,
    opt);
        for(i=0;i<MAX_LINE;i++) free(Inp[i]);
        free(Inp); Inp = NULL;
        return aux;
      }
    }
  }
  return WRONG;

}

int   FGetTimeInfo(char InputFile[MAX_LINE][MAX_CHAR_LINE],
    int user,Planning *pt_plan,TimeInfo *Met)
{
  char helpfile[MAX_PATH_LEN]="";
  /* char **Inp;
   * int i,j; */

  if ((2*strlen(path_sep)+strlen("common")
      +strlen("timeinfo_src.pdf"))>=MAX_PATH_LEN)
    {
      Fprintf(TOSCREEN,"%s{n",error_msg[PATH_TOO_LONG]);
      exit(WRONG);
    }
  /* Inp = malloc(sizeof(char *)*MAX_LINE);
   * for(i=0;i<MAX_LINE;i++){
   *   Inp[i]= malloc(sizeof(char)*(strlen(InputFile[i])+1)
    );
   *   for(j=0;j<(signed)strlen(InputFile[i]);j++)
   *     Inp[i][j]=InputFile[i][j];
   *   Inp[i][strlen(InputFile[i])]='{0';
   * } */

  strcpy(helpfile,premiamandir);
  strcat(helpfile,path_sep);
  strcat(helpfile,"common");
  strcat(helpfile,path_sep);
  strcat(helpfile,"timeinfo_src.pdf");

  if (pt_plan->Action=='p')
    {
      (Met->Init)(Met);

      if (user==TOSCREEN)
        {

          Met->Par[0].Val.V_INT=WRONG;
          return OK;

        }

      return ShowTimeInfo(TOSCREENANDFILE,pt_plan,Met);
    }
  else
    return OK;
}


int FSelectTest(char InputFile[MAX_LINE][MAX_CHAR_LINE],
    int user,Planning *pt_plan,
                Pricing *pt_pricing, Option *opt,Model *
    mod,PricingMethod *met,DynamicTest **test)
{
  int i,ii,isub,k,j;
  DynamicTest** list;
  DynamicTest* dummy;
  char line[MAX_CHAR_LINE];
  if (pt_plan->Action=='t')
    {
      Fprintf(TOSCREEN,"{n_______________________TEST CHOIC
    E:{n{n");
      list=pt_pricing->Test;
      i=0;isub=0;
      dummy=*list;
      while ( dummy !=NULL)
        {
          if( (dummy->CheckTest)(opt,mod,met) == OK)
            {
              for(ii=0;ii<MAX_LINE;ii++){
                j=0;
                while(j<=(signed)(strlen(InputFile[ii])-
    strlen((pt_pricing->Test[i])->Name)))
                  {
                    if( (j==0 || (j>0 && InputFile[i][j-1]=
    =' ')) &&
                        ( InputFile[i][j+strlen((pt_pricing
    ->Test[i])->Name)] == ' ' ||
                          InputFile[i][j+strlen((pt_pricing
    ->Test[i])->Name)] == '{0')
                      ){
                      for(k=j;k<j+(signed)strlen((pt_prici
    ng->Test[i])->Name);k++)
                        line[k-j] = InputFile[ii][k];
                      line[j+(signed)strlen((pt_pricing->
    Test[i])->Name)]='{0';
                      if (StrCasecmp((pt_pricing->Test[i])-
    >Name,line) == 0){
                        *test=pt_pricing->Test[i];
                        (*test)->Init(*test, opt);
                        goto ok;
                      }
                    }
                    j++;
                  }
              }
              isub++;
            }
          i=i+1;
          list++;dummy=*list;
        }


      if (isub==0)
        {
          Fprintf(TOSCREEN,"No test available!{n");
        }
      else
        {
          Fprintf(TOSCREEN,"No tests found, exiting....{n")
    ;
        }
    }
ok:
  return OK;
}
```

References