```c
#include "display.h"

#define NB_SEPAR 21
#define NB_ESP 3
int INDIC_BALISE[6]={0};
int LASTBAL=-1;
char NOM_FICHIER_AMANIP[256]="";
char NOM_INDEX_AMANIP[256]="";
int LASTLINE=-1;
int LENUMLIG=1;

void getNomFich(char nom[])
{
    strcpy(nom,NOM_FICHIER_AMANIP);
}
void getNomIndex(char nom[])
{
    strcpy(nom,NOM_INDEX_AMANIP);
}

void setNomFich(char *nom)
{
    if(nom)
        strcpy(NOM_FICHIER_AMANIP,nom);
}
void setNomIndex(char *nom)
{
    if(nom)
        strcpy(NOM_INDEX_AMANIP,nom);
}
char separ[NB_SEPAR]={' ','\t','\n','/','\\','+',';',',','!','?','(',')','[',']',
'@','%','-','_','\'','\"',EOF};
char espac[NB_ESP]={' ','\n','\t'};
int separateur(char c)
{
    int i=0;
    while(i<NB_SEPAR && separ[i]!=c)
    {
        i++;
    }
    return (i<NB_SEPAR);
}
int espacement(char c)
{
    int i=0;
    while(i<NB_ESP && espac[i]!=c)
    {
        i++;
    }
    return (i<NB_SEPAR);
}
void ClearChaine(char *chaine,int longu)
{
    int i=0;
    for(i=0;i<longu;i++)
        chaine[i]='\0';
}
void color(int t,int f)
{
        HANDLE H=GetStdHandle(STD_OUTPUT_HANDLE);
            SetConsoleTextAttribute(H,f*16+t);
}
void gotoxy (int x, int y)
{
    COORD pos;
    pos.X = x;
```

```
66      pos.Y = y;
67      HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
68      if (!SetConsoleCursorPosition(hConsole, pos))
69      {
70          printf("Erreur dans le GOTOXY CURSOR POS SET");
71      }
72  }
73  void DrawBox(int x,int y,int sizeX,int sizeY)
74  {
75      int i=x;
76      gotoxy(x,y);printf("%c",201);
77      for(i=x+1;i<x+sizeX;i++)  printf("%c",205);
78      printf("%c",187);
79
80      gotoxy(x,y+sizeY);printf("%c",200);
81      for(i=x+1;i<x+sizeX;i++)  printf("%c",205);
82      printf("%c",188);
83
84      for(i=y+1;i<y+sizeY;i++)
85      {
86          gotoxy(x,i);
87          printf("%c",186);
88          gotoxy(x+sizeX,i);
89          printf("%c",186);
90      }
91      gotoxy(x+1,y+1);printf("%c",177);
92      gotoxy(x+1,y+sizeY-1);printf("%c",177);
93      gotoxy(x+sizeX-1,y+1);printf("%c",177);
94      gotoxy(x+sizeX-1,y+sizeY-1);printf("%c",177);
95  }
96  void EraseZone(int x,int y,int sizeX,int sizeY)
97  {
98      int i=x,j=y;
99      for(j=y;j<y+sizeY;j++)
100     {
101         i=x;
102         gotoxy(i,j);
103         for(i=x;i<x+sizeX;i++)
104         {
105             printf(" ");
106         }
107     }
108 }
109 void suppSpace(char *chaine)
110 {
111     char tmpCh[200]="";
112     ClearChaine(tmpCh,200);
113     int i=0;
114     while((chaine[i]==' '))
115     {
116         i++;
117     }
118     int j=0;
119     while(i<strlen(chaine))
120     {
121         tmpCh[j]=chaine[i];
122         i++;
123         j++;
124     }
125     tmpCh[j]='\0';
126     i=strlen(tmpCh)-1;
127     while((tmpCh[i]==' '))
128     {
129         tmpCh[i]='\0';
130         i--;
131     }
```

```
132     strcpy(chaine,tmpCh);
133 }
134 /** MOTION FICHIERS */
135 /// ME PLACE AU DEBUT DE LA LIGNE @line
136 char * GotoLine(FILE *f,int page,int line)
137 {
138     int numChar=0;
139     char * chaine= malloc(200*sizeof(chaine));
140     int c=0,ind=0,nbLig=1,nbCar=0,bal=-1;
141     ClearChaine(chaine,200);
142     if(f)
143     {
144         rewind(f);
145         GotoZePage(f,page);
146         while(c!=EOF && nbLig < line)
147         {
148                 bal=fonc_dec_balise(f,&numChar);
149                 c=fgetc(f);
150                 if(c!='\n')
151                 {
152                     chaine[ind]=c;
153                     ind++;
154                 }
155 
156                 nbCar++;
157                 if(c=='\n' || nbCar>NB_CARAC_LIG_MAX || c==EOF || bal==4 || nbLig>
NB_LIG_PAGE_MAX || (bal>=0 && LASTBAL>=0))
158                 {
159                     nbCar=0;
160                     nbLig++;
161                     arrangeLigne(f,chaine,&numChar);
162 
163                     ind=0;
164 
165                 }
166 
167                 if(bal==4 || nbLig>NB_LIG_PAGE_MAX)
168                 {
169                     /*gotoxy(15,4+nbLig);
170                     printf("%s\n",chaine);*/
171                     nbCar=0;
172                     nbLig=1;
173                     ind=0;
174 
175                 }
176                 else if (bal>=0)
177                 {
178                     //INDIC_BALISE[bal]=!INDIC_BALISE[bal];
179                 }
180         }
181     }
182     return chaine;
183 }
184 /** FONCTIONS AFFICHAGE */
185 int fonc_dec_balise(FILE * f,int * numChar)
186 {
187 
188     int out=-1;
189     int c=0;
190     LASTBAL=-1;
191     c=fgetc(f);
192     (*numChar)++;
193     if(c=='/')
194     {
195         c=fgetc(f);
196         (*numChar)++;
```

```
197         switch(c)
198         {
199             case 'c': out=0;
200                 break;
201             case 'd': out=1;
202                 break;
203             case 'g': out=2;
204                 break;
205             case '-': out=3;
206                 break;
207             case 'p': out=4;
208                 break;
209         }
210         c=fgetc(f);
211         (*numChar)++;
212         if(c!='/' && c!=EOF)
213         {
214             out=-1;
215             (*numChar)-=3;
216             fseek(f,-3,SEEK_CUR);
217         }
218         else if (c!=EOF && out >=0)
219         {
220             if(INDIC_BALISE[out])
221                 LASTBAL=out;
222             INDIC_BALISE[out]=!INDIC_BALISE[out];
223         }
224 
225     }
226     else if (c!=EOF)
227     {
228         fseek(f,-1,SEEK_CUR);
229         (*numChar)--;
230     }
231 
232 
233     return out;
234 }
235 int traiter_dep_mort(FILE *f,char *chaine)
236 {
237     int numChar=0; // USELESS
238     int c=0;
239     int i=0,j=0;
240     int lon=strlen(chaine);
241     int out=0;
242     if (f&& ! espacement(chaine[lon-1]))
243     {
244         c=fgetc(f);
245         while(!separateur(c))
246         {
247             fseek(f,-2,SEEK_CUR);
248             c=fgetc(f);
249             i++;
250         }
251         while(j<i-1)
252         {
253             chaine[lon-1-j]='\0';
254             j++;
255         }
256         i=0;
257         int bal=-1;
258         while(espacement(c))
259             c=fgetc(f);
260         while((!separateur(c) && bal<0) || c==' ')
261         {
262             c=fgetc(f);
```

```
263             bal=fonc_dec_balise(f,&numChar);
264             i++;
265         }
266         if(c=='\n' && bal<0)
267             out=1;
268         else
269             out=0;
270         fseek(f,-i,SEEK_CUR);
271 
272 
273         c=fgetc(f);
274         if(!espacement(c))
275             fseek(f,-1,SEEK_CUR);
276     }
277     return out;
278 }
279 void GotoZeChar(char *nom_fich, int leChar, int *PAGE, int *LINE )
280 {
281     int ledep=0;
282     int numChar = 0;
283     FILE *f= fopen(nom_fich,"r");
284     int compt_Pages=-1,nbLig=-1;
285     if(f)
286     {
287         int ind=0,nbCar=0,c=0,bal=-1;
288         char chaine[200];
289         compt_Pages=1;
290         nbLig=1;
291         ClearChaine(chaine,200);
292         while(!feof(f) && numChar<leChar)
293         {
294                 bal=fonc_dec_balise(f,&numChar);
295                 c=fgetc(f);
296                 numChar++;
297                 if(c!='\n')
298                 {
299                     chaine[ind]=c;
300                     ind++;
301                 }
302 
303                 nbCar++;
304                 if(c=='\n' || nbCar>NB_CARAC_LIG_MAX)
305                 {
306                     nbCar=0;
307                     nbLig++;
308                     ledep=arrangeLigne(f,chaine,&numChar);
309 
310                     //system("pause");
311                     ind=0;
312                     c='\0';
313                     ClearChaine(chaine,200);
314                 }
315 
316                 if(bal==4 || nbLig>=NB_LIG_PAGE_MAX)
317                 {
318                     nbCar=0;
319                     nbLig=1;
320                     compt_Pages++;
321                 }
322                 else if (bal>=0)
323                 {
324                     //INDIC_BALISE[bal]=!INDIC_BALISE[bal];
325                 }
326         }
327         fclose(f);
328     }
```

```
329     *PAGE=compt_Pages;
330     *LINE = nbLig+1; // RETOURNE LA PAGE OU SE TROUVE LE CHAR
331     /*if(ledep)
332         (*LINE)++; */
333 }
334 void GotoZePage(FILE *f,int page)
335 {
336     int numChar=0; // USELESS
337     if(f)
338     {
339         rewind(f);
340         int killDatSlashN=0,ind=0,nbCar=0,nbLig=1,compt_Pages=1,c=0,bal=-1;
341         char chaine[200];
342         ClearChaine(chaine,200);
343         while(!feof(f) && compt_Pages < page)
344         {
345                 bal=fonc_dec_balise(f,&numChar);
346                 c=fgetc(f);
347                 if(c!='\n')
348                 {
349                     chaine[ind]=c;
350                     ind++;
351                 }
352 
353                 nbCar++;
354                 if(c=='\n' || nbCar>NB_CARAC_LIG_MAX)
355                 {
356                     nbCar=0;
357                     nbLig++;
358                     killDatSlashN=arrangeLigne(f,chaine,&numChar);
359                     ind=0;
360                     c='\0';
361                     ClearChaine(chaine,200);
362                 }
363 
364                 if(bal==4 || nbLig>=NB_LIG_PAGE_MAX)
365                 {
366                     nbCar=0;
367                     nbLig=1;
368                     compt_Pages++;
369                 }
370                 else if (bal>=0)
371                 {
372                     //INDIC_BALISE[bal]=!INDIC_BALISE[bal];
373                 }
374         }
375 
376     }
377 }
378 int arrangeLigne(FILE *f,char *chaine,int *numChar)
379 {
380     int c=0;
381     int i=0,j=0;
382     int lon=strlen(chaine);
383     int out=0;
384     if (f)
385     {
386         c=fgetc(f);
387         (*numChar)++;
388         // REVENIR AU PREMIER SEPARATEUR
389         if(separateur(c))
390         {
391             fseek(f,-1,SEEK_CUR);
392             (*numChar)--;
393         }
394         while(!separateur(c) || c=='\'' )
```

```
395         {
396             fseek(f,-2,SEEK_CUR);
397             c=fgetc(f);
398             (*numChar)--;
399             i++;
400         }
401         // REMPLACER PAR DES \0 CE QUI A DEJA ETE ECRIT
402         if(i>0)
403             out=1;
404         while(j<i-1)
405         {
406             chaine[lon-1-j]='\0';
407             j++;
408         }
409         i=0;
410         int bal=-1;
411         //while(espacement(c))
412             //c=fgetc(f);
413         /*while((!separateur(c) && bal<0)&& c!=EOF)
414         {
415             bal=fonc_dec_balise(f,numChar);
416             c=fgetc(f);
417             (*numChar)++;
418 
419             i++;
420         }
421         /*if(c=='\n' && bal<0)
422             out=1;
423         else
424             out=0;
425         fseek(f,-i-2,SEEK_CUR);
426         (*numChar)-=i+2;
427 
428 
429         c=fgetc(f);
430         (*numChar)++;
431         if(!espacement(c))
432         {
433             fseek(f,-1,SEEK_CUR);
434             (*numChar)--;
435         } */
436 
437     }
438     if(chaine[strlen(chaine)-1]=='/')
439         chaine[strlen(chaine)-1]='\0';
440 
441     return out;
442 }
443 int NbWords(char * nom_fich)
444 {
445     int numChar=0;
446     int cpt=0,c=0,bal=-1;
447     FILE *f=NULL;
448     f=fopen(nom_fich,"r");
449     char *word=NULL;
450     if(f)
451     {
452         while(!feof(f))
453         {
454             word=getNextWord(f,&numChar);
455 
456             if(word)
457             {
458                 suppSpace(word);
459                 if(word[0]!='\0')
460                     cpt++;
```

```
                free(word);
            }
        }
    }
    fclose(f);
    if(cpt!=0)
        cpt++;
    fclose(f);
    return cpt;
}
int nbPages(char *nom_fichier)
{
    FILE *f = fopen(nom_fichier,"r");
    int compt_Pages=1;
    if(f)
    {
        rewind(f);
        int killDatSlashN=0,ind=0,nbCar=0,nbLig=1,c=0,bal=-1;
        int numChar=0;
        char chaine[200];
        ClearChaine(chaine,200);
        while(!feof(f))
        {
                bal=fonc_dec_balise(f,&numChar);
                c=fgetc(f);
                if(c!='\n')
                {
                    chaine[ind]=c;
                    ind++;
                }

                nbCar++;
                if(c=='\n' || nbCar>NB_CARAC_LIG_MAX)
                {
                    nbCar=0;
                    nbLig++;
                    killDatSlashN=arrangeLigne(f,chaine,&numChar);
                    ind=0;
                    c='\0';
                    ClearChaine(chaine,200);
                }

                if(bal==4 || nbLig>=NB_LIG_PAGE_MAX)
                {
                    nbCar=0;
                    nbLig=1;
                    compt_Pages++;
                }
                else if (bal>=0)
                {
                    //INDIC_BALISE[bal]=!INDIC_BALISE[bal];
                }
        }

    }
    fclose(f);
    return compt_Pages;
}
int nbLig(char *nom_fichier)
{
    FILE *f = fopen(nom_fichier,"r");
    int compt_Pages=1;
    if(f)
    {
        rewind(f);
        int killDatSlashN=0,ind=0,nbCar=0,nbLig=1,c=0,bal=-1;
```

```
527         int numChar=0;
528         char chaine[200];
529         ClearChaine(chaine,200);
530         while(!feof(f))
531         {
532                 bal=fonc_dec_balise(f,&numChar);
533                 c=fgetc(f);
534                 if(c!='\n')
535                 {
536                     chaine[ind]=c;
537                     ind++;
538                 }
539 
540                 nbCar++;
541                 if(c=='\n' || nbCar>NB_CARAC_LIG_MAX)
542                 {
543                     nbCar=0;
544                     nbLig++;
545                     killDatSlashN=arrangeLigne(f,chaine,&numChar);
546                     ind=0;
547                     c='\0';
548                     ClearChaine(chaine,200);
549                 }
550 
551                 if(bal==4 || nbLig>=NB_LIG_PAGE_MAX)
552                 {
553                     nbCar=0;
554                     nbLig=1;
555                     compt_Pages++;
556                 }
557                 else if (bal>=0)
558                 {
559                     //INDIC_BALISE[bal]=!INDIC_BALISE[bal];
560                 }
561         }
562 
563     }
564     fclose(f);
565     return compt_Pages;
566 }
567 int premPosWordChaine(char *chaine,char*mot,int myem)
568 {
569     int i=0,j=0,save=0,yem=0;
570     char word[256]={'\0'};
571     while( chaine[i]!='\0' && yem<myem)
572     {
573         save=i;
574         while(!separateur(chaine[i]) && chaine[i]!='.' && chaine[i]!='\0')
575         {
576             word[j]=chaine[i];
577             j++;
578             i++;
579 
580         }
581 
582         word[j]='\0';
583 //          printf(":%s:",word);
584         if(!strcmp(word,mot))
585         {
586             yem++;
587         }
588         while(separateur(chaine[i]) || chaine[i]=='.' && chaine[i]!='\0')
589         {
590             i++;
591         }
592         j=0;
```

```
593
594
595         }
596         if(yem && save!=0)
597             return save-1;
598         else if (yem && save==0)
599             return -1;
600         else
601             return -2;
602     }
603     void AffichePage(char *nom_fichier,int page, int MODE , int LINE,char *mot)
604     {
605         color(15,0);
606         int numChar=0;
607         int rein=0;
608
609         while(rein<6)
610         {
611             INDIC_BALISE[rein]=0;
612             rein++;
613         }
614         FILE *f=NULL;
615         int bal=-1;
616         f=fopen(nom_fichier,"r");
617         if(f)
618         {
619             strcpy(NOM_FICHIER_AMANIP,nom_fichier);
620
621             //printf("%d\n",NbWords(f));
622             // system("pause");
623             GotoZePage(f,page);
624             int CBON=0;
625             int killDatSlashN=0,ind=0,nbCar=0,nbLig=1,compt_Pages=1,c=0,bal=-1;
626             char chaine[200];
627             ClearChaine(chaine,200);
628             char *tempChaine=malloc(200/*NB_CARAC_LIG_MAX*2*sizeof(char)*/);
629             ClearChaine(tempChaine,200);
630             while(c!=EOF && nbLig < NB_LIG_PAGE_MAX)
631             {
632                 color(15,0);
633                     bal=fonc_dec_balise(f,&numChar);
634                     c=fgetc(f);
635                     if(c!='\n')
636                     {
637                         chaine[ind]=c;
638                         ind++;
639                     }
640
641                     nbCar++;
642                     if(c=='\n' || nbCar>NB_CARAC_LIG_MAX || c==EOF || bal==4 || nbLig>
NB_LIG_PAGE_MAX || (bal>=0 && LASTBAL>=0))
643                     {
644                         nbCar=0;
645                         nbLig++;
646                         //printf("AV:%s\n",chaine);
647                         killDatSlashN=arrangeLigne(f,chaine,&numChar);
648                         gotoxy(15,4+nbLig);
649                         /** TRAITEMENT SELON LA BALISE*/
650                         int longueur =strlen(chaine);
651                        /* int parc=0;
652                         while(parc<5)
653                         {
654
655                             printf("%d",INDIC_BALISE[parc]);
656                             parc++;
657                         }*/
```

```
                    int i=0;
                    if(INDIC_BALISE[0] ||LASTBAL ==0)
                    {
                        strcpy(tempChaine,chaine);
                        i=0;
                        while(2*i<NB_CARAC_LIG_MAX-longueur)
                        {
                            chaine[i]=' ';
                            i++;
                        }
                        chaine[i]='\0';
                        i=0;
                        strcat(chaine,tempChaine);
                    }
                    else if(INDIC_BALISE[1]||LASTBAL ==1) /** FIXER QUAND BALISE
DANS LIGNE PREND LA LIGNE A DROITE FIXER EN METTANT UN SAUT DE LIGNE ET TRAITER SEPAR */
                    {
                        i=0;
                        suppSpace(chaine);
                        longueur=strlen(chaine);
                        strcpy(tempChaine,chaine);
                        while(i<NB_CARAC_LIG_MAX-longueur+1)
                        {
                            chaine[i]=' ';
                            i++;
                        }
                        chaine[i]='\0';

                        strcat(chaine,tempChaine);
                    }
                    else if (INDIC_BALISE[2]||LASTBAL ==2)
                    {
                        suppSpace(chaine);
                    }
                    else if (INDIC_BALISE[3]||LASTBAL ==3)
                    {
                        // voir apres pr jutifie
                    }
                    else
                    {
                        suppSpace(chaine);
                    }
                    char sauvegarde[300]="";
                    char Part1[256]="";
                    char Part2[256]="";
                    int kifkif=0;
                    if (MODE ==2 && nbLig==LINE)
                    {
                        if(LASTLINE==-1)
                            LASTLINE=LINE;
                        else if(LASTLINE==LINE)
                            LENUMLIG++;
                        else
                        {
                            LASTLINE=LINE;
                            LENUMLIG=1;
                        }


                        strcpy(sauvegarde,chaine);
                        color(0,15);

                        char *mots=strtok(sauvegarde," \n./\\+;,!?()[]@%-_'");
                        while(mots && ! kifkif)
                        {
                            //printf("%s",mots);
```

```
723
724                             //getch();
725                             if(mots!=NULL && !strcmp(mots,mot))
726                             {
727                                 kifkif=1;
728                             }
729                             mots = strtok(NULL," \n./\\+;,!?()[]@%-_'");
730
731                         }
732                         MODE=-1;
733                         if(kifkif==0)
734                         {
735                             color(15,0);
736                             LINE++;
737                             MODE=2;
738                             CBON=0;
739                         }
740                         else
741                         {
742                             int loch=strlen(chaine);
743                             int pos=premPosWordChaine(chaine,mot,LENUMLIG);
744                             int i=0;
745                             while(i<=pos)
746                             {
747                                 Part1[i]=chaine[i];
748                                 i++;
749                             }
750                             Part1[i]='\0';
751                             i=i+strlen(mot);
752                             int j=0;
753                             while(i<loch)
754                             {
755                                 Part2[j]=chaine[i];
756                                 i++;
757                                 j++;
758                             }
759                             Part2[j]='\0';
760                             CBON=1;
761
762                         }
763
764                     }
765                     if(CBON )
766                     {
767                         color(15,0);
768                         printf("%s",Part1);
769                         color(0,15);
770                         printf("%s",mot);
771                         color(15,0);
772                         printf("%s",Part2);
773                         CBON=0;
774                     }
775                     else
776                         printf("%s\n",chaine);
777
778                     if(MODE ==-1)
779                     {
780                         color(15,0);
781                         MODE=0;
782                     }
783
784                     //system("pause");
785                     ind=0;
786                     ClearChaine(chaine,200);
787                 }
788
```

```
789                 if(bal==4 || nbLig>NB_LIG_PAGE_MAX)
790                 {
791                     /*gotoxy(15,4+nbLig);
792                     printf("%s\n",chaine);*/
793                     nbCar=0;
794                     nbLig=1;
795                     compt_Pages++;
796 
797                 }
798                 else if (bal>=0)
799                 {
800                     //INDIC_BALISE[bal]=!INDIC_BALISE[bal];
801                 }
802         }
803         color(0,12);
804         gotoxy(POS_X+NB_CARAC_LIG_MAX/2+ 7, POS_Y + NB_LIG_PAGE_MAX +3);
805         printf("%d   /    ",page);
806         color(15,0);
807         free(tempChaine);
808         fclose(f);
809     }
810 
811 
812 
813 }
814 /** FONCTIONS INDEX*/
815 char * getNextWord(FILE *f,int *numChar)
816 {
817     char *word=malloc(200*sizeof(char));
818     ClearChaine(word,200);
819     int c=0,i=0;
820     fonc_dec_balise(f,numChar);
821     while(separateur(c) && c!=EOF)
822     {
823         c=fgetc(f);
824         (*numChar)++;
825     }
826 
827     while(!separateur(c) && c!='.' && c!=EOF)
828     {
829         if(c!=0)
830         {
831             fonc_dec_balise(f,numChar);
832             word[i]=c;
833             i++;
834         }
835         c=fgetc(f);
836         (*numChar)++;
837     }
838     word[i]='\0';
839 
840     if(c==EOF)
841         return NULL;
842 
843     return word;
844 
845 }
846 int getNextNumbLine(FILE *f)
847 {
848     int n=0;
849     char c=0;
850     while(!(  (c>='0')&& (c<='9')) && c!=EOF && c!='\n')
851     {
852         c=fgetc(f);
853     }
854     while((c>='0')&& (c<='9') && c!=EOF && c!='\n')
```

```
855      {
856          n=n*10+c-'0';
857          c=fgetc(f);
858      }
859      if(c==EOF)
860          return -1;
861      return n;
862  }
863  /**char *file_to_tab(FILE *f)
864  {
865      #define NB_char 20
866      char *chaine;
867      char *token;
868      char separ[NB_SEPAR]={'
',',','\n','/',,'\\','+',';',',','!','?','(',')','[',']','@','%','-','_','\'','\"',EOF};
869      chaine=malloc(NB_char*sizeof(char));
870      int i=0;
871      while(!feof(f) && i<NB_char)
872      {
873          chaine[i]=fgetc(f);
874          i++;
875      }
876      token=strtok(chaine,separ);
877      while(token!=NULL)
878      {
879          printf("%s\n",chaine);
880          token=strtok(NULL,separ);
881      }
882
883      return chaine;
884
885  }*/
886  /*typedef struct
887  {
888      int page;
889      int ligne;
890      int posMot;
891  } position;
892  typedef struct
893  {
894      char *mot;
895      position *pos;
896  } mot;*/
897  void init_index(char *nom_fich)
898  {
899      /**D*/
900      strcpy(NOM_FICHIER_AMANIP,nom_fich);
901
902      char nomInd[256]="";
903      int numChar=0,compt_pages=1,numMot=0;
904      int longueur= strlen(nom_fich);
905      int POS_SAVE=0;
906      char *mot;
907      FILE *f=NULL,*ind = NULL;
908      /**C*/
909      strcpy(nomInd,nom_fich);
910      nomInd[longueur-4]='I';
911      nomInd[longueur-3]='N';
912      nomInd[longueur-2]='D';
913      nomInd[longueur-1]='.';
914      nomInd[longueur]='t';
915      nomInd[longueur+1]='x';
916      nomInd[longueur+2]='t';
917      nomInd[longueur+3]='\0';
918      setNomIndex(nomInd);
919      f=fopen(nom_fich,"r");
```

```
920      ind= fopen(nomInd,"w");
921      if(f&& ind)
922      {
923          while(!feof(f))
924          {
925              //POS_SAVE=numChar;
926              mot=getNextWord(f,&numChar);
927              if(mot!=NULL)
928                  if(strlen(mot)>0)
929                  {
930                      numMot++;
931                      fprintf(ind,"%s %d\n",mot,numChar-strlen(mot)-1);
932                  }
933              if(mot!=NULL)
934                  free(mot);
935          }
936      }
937      fclose(ind);
938      fclose(f);
939
940  }
941  int CritereRecherche(char *chaine,char *autre, int approxOUPAS)
942  {
943      if(chaine && autre)
944      {
945          //printf("\nNONULL");
946          if (approxOUPAS)
947          {
948              //printf("\nAVPROX");
949              int tmp=EqAprrox(chaine,autre);
950              //printf("\nAPPROX");
951              return tmp;
952          }
953          else
954          {
955              return !strcmp(chaine,autre);
956          }
957      }
958
959      else
960          return 0;
961  }
962  int recherche_index(char *nom_fich,char *mot,int occNumb,int APPROX)
963  {
964      int numChar=0;
965      int n=0;
966      char *word=NULL;
967      FILE *f=NULL;
968      int nb=0;
969      f= fopen(nom_fich,"r+");
970      int lastPos=0;
971      int positions = -1;
972      if(f)
973          while(!feof(f) && n>=0 && nb<occNumb) //&& positions<0
974          {
975              word=getNextWord(f,&numChar);
976              if(word)
977              {
978                  n=getNextNumbLine(f);
979                  //printf("\nAV CRIT");
980                  if(CritereRecherche(word,mot,APPROX))
981                      {
982                          //printf("tours");
983                          //printf("DONCRIT");
984                          if(n>=0)
985                          {
```

```
 986                             lastPos=positions;
 987                             positions = n;
 988                             nb++;
 989                         }
 990                     }
 991                 //printf("\nAP CRIT");
 992             }
 993             free(word);
 994         }
 995     if(nb< occNumb) //si
 996         positions=-1;
 997     fclose(f);
 998
 999     return positions;
1000 }
1001
1002 int RemplacerUnConf(char *nom_fich,char *mot,char *motRemp,int mYemMot,int
CONFONOFF,int APPROX)
1003 {
1004     color(15,0);
1005     FILE *newF=NULL,*f=NULL;
1006     int longueur=strlen(mot);
1007     int charPos= recherche_index(NOM_INDEX_AMANIP,mot,mYemMot,APPROX);
1008     //printf("LECHARPOS : %d //",charPos);
1009     int mov=0;
1010     if(charPos>=0)
1011     {
1012         if(CONFONOFF)
1013         {
1014             int ligne=0,page=0;
1015             GotoZeChar(nom_fich,charPos,&page,&ligne);
1016             if(charPos<0)
1017                 ligne=-1;
1018             EraseZone(15,5,NB_CARAC_LIG_MAX+3,NB_LIG_PAGE_MAX);
1019             AffichePage(nom_fich,page,2,ligne,mot);
1020             /*char *leword =getMotCharPos(nom_fich,charPos);
1021             printf("REPLACE THIS MATCH ? %s",leword);
1022             free(leword); */
1023             mov=getch();
1024         }
1025         else
1026             mov=13;
1027
1028
1029
1030         if(mov==13)
1031         {
1032             newF=fopen("tmp.txt","w+");
1033             f=fopen(nom_fich,"r");
1034             rewind(f);
1035             int i=0,c=0;
1036             while(i<charPos&& c!=EOF)
1037             {
1038                 c=fgetc(f);
1039                 fprintf(newF,"%c",c);
1040                 i++;
1041             }
1042             i=0;
1043             while(i<longueur)
1044             {
1045                 c=fgetc(f);
1046                 i++;
1047             }
1048             fprintf(newF,"%s",motRemp);
1049             while(c!=EOF)
1050             {
```

```
1051                c=fgetc(f);
1052                fprintf(newF,"%c",c);
1053            }
1054            fclose(newF);
1055            fclose(f);
1056            rename(NOM_FICHIER_AMANIP,"del.txt");
1057            printf("%s %s",NOM_FICHIER_AMANIP,NOM_INDEX_AMANIP);
1058            //system("rename tes.txt del.txt");
1059            //system("rename tmp.txt tes.txt");
1060            remove("del.txt");
1061            rename("tmp.txt",NOM_FICHIER_AMANIP);
1062            init_index(NOM_FICHIER_AMANIP);
1063
1064            return 1;
1065
1066        }
1067        else if (mov !=27)
1068            return -1;
1069        else
1070            return 0;
1071
1072
1073    }
1074    else
1075    {
1076        printf("IL ES PETIT");
1077        return 0;
1078    }
1079
1080    // PASSER MAYBE NOM DU FICHIER
1081
1082 //    printf("CLOSED : :%s:",NOM_FICHIER_AMANIP);
1083
1084 }
1085 void RemplacerOnebyOne(char *nom_fich,char *mot,char *motRemp,int APPROX)
1086 {
1087     int still=1;
1088     int mYem=1;
1089     if(nom_fich)
1090     {
1091         while(still)
1092         {
1093             still=RemplacerUnConf(nom_fich,mot,motRemp,mYem,1,APPROX); // APPEL
AVEC CONFIRMATION ON
1094             if(still<0)
1095                 mYem++;
1096         }
1097         EraseZone(15,5,NB_CARAC_LIG_MAX+3,NB_LIG_PAGE_MAX);
1098
1099     }
1100 }
1101
1102 void RemplacerTout(char *nom_fich,char *mot,char *motRemp,int APPROX)
1103 {
1104     int still=1;
1105     int myem=0;
1106     if(nom_fich)
1107     {
1108         while(still)
1109         {
1110             still=RemplacerUnConf(nom_fich,mot,motRemp,1,0,APPROX); // APPEL AVEC
CONFIRMATION OFF
1111         }
1112         EraseZone(15,5,NB_CARAC_LIG_MAX+3,NB_LIG_PAGE_MAX);
1113
1114     }
```

```
}

int DisLevenshtein(char *mot,char *mot2)
{
    int cost=0;
    int l1=strlen(mot)+1;
    int l2=strlen(mot2)+1;
    int **tab=malloc(l1*sizeof(int*));
    int i=0,j=0;
    while(i<l1)
    {
        tab[i]=malloc(l2*sizeof(int));
        i++;
    }
    //printf("\nALLOC");
    for(i=0;i<l1;i++)
    {
        tab[i][0]=i;
    }
    for(j=0;j<l2;j++)
    {
        tab[0][j]=j;
    }
    for(i=1;i<l1;i++)
    {
        for(j=1;j<l2;j++)
        {
            if(mot[i-1]==mot2[j-1])
                cost=0;
            else
                cost=1;
            tab[i][j]=min(tab[i-1][j]+1, min(tab[i][j-1]+1,tab[i-1][j-1]+cost));
        }
    }
    //printf("\nCALCUL");
    int leret=tab[l1-1][l2-1];
    //printf("\FINC");
    i=0;
    while(i<l1)
    {
        if(tab[i])
            free(tab[i]);
        tab[i]=NULL;
        i++;
    }
    if(tab!=NULL)
        free(tab);
    //printf("\nLib");
    return leret;
}
/*void tri_bulles(char * tab)
{
    int ord = 0;
    int taille = strlen(tab);
    while(!ord)
    {
        ord = 1;
        for(int i=0 ; i < taille-1 ; i++)
        {
            if(tab[i] > tab[i+1])
            {
                swap(tab[i],tab[i+1]);
                ord = 0;
            }
        }
        taille--;
```

```
1181      }
1182  }
1183  int sameLetters(char *mot, char mot2)
1184  {
1185      int l1=strlen(mot);
1186      int l2=strlen(mot2);
1187      if(l1<l2)
1188      {
1189
1190      }
1191  }*/
1192  int LettreInvers(char *ch,char *autre)
1193  {
1194      int resu=0;
1195      char chaine[256]="";
1196      strcpy(chaine,ch);
1197      int i=0, lon=strlen(chaine);
1198      char tmp=0;
1199      if (abs(lon-strlen(autre))<=1)
1200      {
1201          while(i<lon-1 && !resu)
1202          {
1203              tmp=chaine[i];
1204              chaine[i]=chaine[i+1];
1205              chaine[i+1]=tmp;
1206              if(DisLevenshtein(chaine,autre)<2)
1207              {
1208                  resu=1;
1209              }
1210              tmp=chaine[i];
1211              chaine[i]=chaine[i+1];
1212              chaine[i+1]=tmp;
1213              i++;
1214          }
1215
1216      }
1217      return resu;
1218
1219  }
1220  int EqAprrox(char *mot,char *mot2)
1221  {
1222      if (mot && mot2)
1223      {
1224          //printf("\nNONONULL");
1225          if(DisLevenshtein(mot,mot2)<2 || LettreInvers(mot,mot2))
1226          {
1227              //printf("L7A9T");
1228              return 1;
1229          }
1230      }
1231
1232      return 0;
1233  }
1234
1235
```