

**PROJECT REPORT (VOLUME II)**

**SPELL CHECKER  
IN  
GURMUKHI**

---

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT  
FOR THE DEGREE OF**

**MASTER OF COMPUTER APPLICATIONS**

**BY**

**ASHWANI KUMAR  
(2/90)**

**NEENA TAYAL  
(13/90)**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY  
PATIALA - 147 001**

**(DEEMED TO BE A UNIVERSITY)**

**MAY 1993**

```

EUR.C */
#include<stdio.h>
#include<string.h>
#include<memory.h>
#include<search.h>
#include<ctype.h>

char c,c1,w[200][20],w1[200][20],w3[40],w4[40],d[10][20],temp[40],w5[10][20],
w6[10][20],w7[20][20],arr[40],ch,*mn;
int i,j=0,s,z,p1,q1,b,m1,n1;
FILE *fp1,*fp,*fp2,*fp3,*fp4,*fp5;
main()

    /* opens the dictionary of words in original order*/
fp=fopen("sdic","r");

    /* opens the dictionary containing mirror images of words */
fp1=fopen("sdic1","r");

    /* opens a temporary file where all the changes to be stored*/
fp3=fopen("tr","w");

    /* a procedure scr is called */
scr();

    /* opens the input file which is to be checked */
fp2=fopen(arr,"r");

    /* a procedure sd is called */
sd();

    /* a procedure sdl is called */
sdl();

    this is a loop which will go on executing until the end of file*/
{
    /* a procedure ab() is called */

```

```

ab();
/* a procedure is pr1 called */
pr1();
/* this enters the string contained in w3 into "tr " file */
fprintf(fp3,"%s",w3);
}
while(!feof(fp2));
/* closes the input file*/
fclose(fp2);
/* closes the "tr" file*/
fclose(fp3);
printf("IF YOU WANT TO SAVE THE CHANGES (Y/N) ");
/* get the choice of user yes or no */
scanf("\n%c", &ch);
/* conditional loop which will be executed only if 'y' is entered by
user*/
if((ch=='Y') || (ch=='y'))
{
/* a procedure named rep is called to replace the original input
file with tr file( to save the changes )*/
rep();
}
else
{
printf("FILE IS NOT CHANGED\n ");
}
printf("WHETHER YOU WANT TO APPEND IN THE DICTIONARY(Y/N) ");
/* it gets user's choice 'y' or 'n'*/
scanf("\n%c", &ch);
printf("\n");

```

```

/* conditional loop which will execute only if choice entered is 'y'*/
if((ch=='Y') || (ch=='y'))
{
    printf("IF NO MORE WORDS TO APPEND, PLEASE ENTER A FULL STOP\n");
    /* a procedure is called to append new words in dictionary */
    append();
}
}

/*****
THIS PROCEDURE PUTS THE DICTIONARY OF PANJABI WORDS
IN AN ARRAY W[] , HERE WORDS ARE IN THEIR ORIGINAL
[BORN AND IN SORTED ORDER.....
*****/

sd()
{
/* initialise i */
i=0;
/* a loop which will execute until end of file reached or
no of words stored in array becomes more than 200*/
while((!feof(fp))&&(i<=200))
{
    j=0;
    /* a loop that will execute until get character is not
equal to carriage return '\n' or a blank ' ' */
while(((c=fgetc(fp))!=10)&&(j<=19))
/* this stores one character at w[i][j] position and j will be
incremented every time */
w[i][j++]=c;
/* each word stored in array is finished with '0'*/
w[i+1][j]=0;
}
}

```

```
/* this stores the no of words that are stored in array */
ml=i;
}
```

```
/******
THIS PROCEDURE TAKES ONE WORD FROM THE INPUT TEXT AND
PUTS IT IN AN ARRAY W3[] , THEN IT REVERSES THE WORD
THEN PUTS IT IN ANOTHER ARRAY W4[].....
******/
```

```
ab()
{
int y,m,a;
/* it sets the memory space for array */
memset(w3,0,40);
a=0;
/* a loop that will store the word in w3 one by one character
and execute until blank ,return or tab comes */

if(((c=fgetc(fp2))!=32)&&(c!=10)&&(c!=8))
{
w3[a++]=c;
/* a loop that will store the word in w3 one by one character
and execute until blank ,return or tab comes */

while((!feof(fp2))&&((c=fgetc(fp2))!=32)&&(c!=10)&&(c!=8))
{
w3[a++]=c; /* puts c in array at position a*/
}

/* puts the character in tr file */
fputc(c,fp3);

/* initialises the array */
w3[a]=0;
}
else
{

/*puts one character in the file, tr */
```

```

    fputc(c,fp3);
}

/* sets memory space of 40 for array w4 */
memset(w4,0,40);
m=0;

/* a loop that stores the reverse of the word in another array
here y is initialised to the length of string in w3 , and is decremented
each time , function strlen returns the length of string */
for(y=strlen(w3)-1;y>=0;y--)
{
    /* a conditional loop that will execute if and only if next position
in array does not contains return or tab or blank */

    if((w3[y]!=' ')&&(w3[y]!='\t')&&(w3[y]!='\n'))
        w4[m++]=w3[y];
}

/*****
THIS PROCEDURE PUTS THE DICTIONARY OF PANJABI WORDS
IN AN ARRAY w1[] ,HERE WORDS ARE IN THEIR REVERSE
FORM AND IN SORTED ORDER.....
*****/

sdl()
{
    i=0;
    /* a loop will execute until eof or no of words stored > 200 */
    while((!feof(fp1))&&(i<=200))
    {
        j=0;
        /* a loop will execute until word is ended */
        while(((c=fgetc(fp1))!=10)&&(j<=19))
            w1[i][j]=c;
        w1[i][j]=0;
    }
}

```

```

/*****
THIS PROCEDURE IS CALLED BY pr1(). THE MIRROR IMAGE OF THE INPUT WORD IS
TAKEN IN THIS PROCEDURE OR THE INPUT WORD IS REVERSED IN THIS PROCEDURE.
AFTER COMPARISON OF THIS WORD WITH REVERSE DICTIONARY, RESULTING WORDS ARE
STORED IN TWO DIMENSIONAL ARRAY d.NOW THESE WORDS ARE REVERSED AND STORED
IN ARRAY w5.....
*****/

```

```

pr()
{
int f,l,n,v,ll,e,k,x,g;
f=0;l=m1-1;
if (w4[0]!=0)
{
/* loop designed to search the word until eof dictionary */
do
{
x=(f+1)/2;

/* this function returns whether two strings are same or not
value s=0 tells strings are same, -1 or 1 are not same */

s=memcmp(w4,w1[x],strlen(w4));
if(s==0)
{
return;
}
if(s==-1)
l=(x-1);
if(s==1)
f=(x+1);
}while((s!=0) && (f<=l));

/* a loop will execute if word is not found and this finds out the
words having same starting characters as that of input word */
if(s!=0)
{
f=0;l=m1;n=1;
do
{
x=(f+1)/2;

/* compares string in w4 and string at w1[x1] and stores

```

```

        result  in variable s which can have values 0,1,-1  */
s=memcmp(w4,wl[x],3);
if(s==0)
{
    v=x;
    while((s==0)&&(v<=(x+3)))
    {
        /* initialises memory space for array d[]  */
        memset(d[n],0,20);

        /* this funtion copies string from one array to another */
        strcpy(d[n],wl[v]);

        /* compares string in w4  and string at wl[x] and stores
           the result in variable s which can have values 0,1,-1*/

        s=memcmp(w4,wl[v],3);
        n++;
        v++;
    }
    p1=n;
}
if(s==-1)
    l=(x-1);
if(s==1)
    f=(x+1);
}while((s!=0) && (f<=l));

/* this loop is applied to change mirror images of option words to
their original form  */
for(k=1;k<=n;k++)
{
    e=0;

    /* l1 is initialised to length of string in array d[k]*/
    for(l1=(strlen(d[k])-1);l1>=0;l1--)
    {

        /* conditional loop that will execute if d[k][l1] is not
           blank or return or tab  */

```

```

        if((d[k][l1]!=' ')&&(d[k][l1]!='\n')&&(d[k][l1]!='\t'))
        {
            w5[k][e++]=d[k][l1];
        }
    }

    /* this ends the word with zero */
    w5[k][e++]=0;
}
}

/*****
THIS PROCEDURE IS BASICALLY DESIGNED TO FIND THE INPUT TEXT WORD
FROM DICTIONARY AND IF IT IS FOUND DISPLAY CORRECT WORD AND
IF IT IS NOT FOUND, FIND OUT SOME WORDS FROM FROM THE DICTIONARY
WHICH HAVE THE SAME STARTING CHARACTERS AND STORE THEM IN AN
ARRAY NAME W6[] , AFTER THAT CALL PROCEDURES THAT DO THE JOBS
LIKE DISPLAYING CHOICES ETC.....
*****/

pr1()
{
int f,l,n,v1,x1,g;
f=0;l=m1-1;
if (w3[0]!=0)
{
    /* aloop executes to search a input word from dictionary */
do
{
    x1=(f+l)/2;

    /* a function that compares the strings at different locations of
memory and the output in variable s values can be 1,0,-1 */

s=memcmp(w3,w[x1],strlen(w3));
if(s==0)
{
    printf("WORD IS CORRECT : ");

```

```

    /* puts the string on screen in array w3 */

    puts(w3);
    printf("\n");
    return;
}
if(s== -1)
l=(x1-1);
if(s==1)
f=(x1+1);
}while((s!=0) && (f<=1));

/* a conditional loop that will execute if word is not found in
dictionary */
if(s!=0)
{
printf("WORD IS NOT IN DICT : ");
printf("%s",w3);
printf(" : ");
printf("%s",w4);
printf("\n\n");
f=0;
l=m1;
n=1;

/* a loop that searches the dictionary for the words which have
same starting characters as that of input word */

do
{
x1=(f+1)/2;
/* compares two strings at diff. memory places and stores
result in variable s */
s=memcmp(w3,w[x1],1);
v1=x1;

/* a loop made to get some words that have same starting
characters as that of input word */
while((s==0) && (v1<(x1+3)))
{
/* initialises memory space for array w6 */

```

```

memset(w6[n],0,20);
/* copies string */
strcpy(w6[n],w[v1]);
/* compares two strings at diff. memory places */
s=memcmp(w3,w[v1],1);
n++;
/* stores the no words stored in array w6 */
ql=n;
vl++;
}
if(s==-1)
    l=(x1-1);
if(s==1)
    f=(x1+1);
}while((s!=0) && (f<=1));
/* calls procedure pr() */
pr();
/* calls procedure put() */
put();
/* calls procedure df() */
df();
/* calls procedure chn() */
chn();
}
}
/*returns to main program */
return;
}

```

```

/*****
THIS IS DESIGNED TO PROVIDE CHOICES TO THE THE USER IF WORD IS
INCORRECT , THIS PROCEDURE BASICALLY STORES THE DIFFERENT CHOICES
CAME FROM PROCEDURE pr() AND pr1() INTO A SINGLE ARRAY w7[]
FROM WHERE USER CAN MAKE CHOICE .....
*****/

```

```

put()
{
int h1=1,h2=1,h3=1,t1=1,t2=1;

/* this loop is designed to store words of w6 in w7 one by one*/
for(h1=1;h1<=q1;h1++)
{
    /* copies string of w6[h1] into w7[h1] */
    strcpy(w7[h1],w6[h1]);
}

/* this loop is designed to store words of w5 in w7 array
after the previous stored words */
for(h2=(q1+1);h2<=(p1+q1);h2++)
{
    /* copies string from w5[h2] into w7[h2] */
    strcpy(w7[h2],w5[h2]);
}
printf("AVAILABLE CHOICES ARE HERE YOU CAN SELECT ANYONE\n");
printf("\n");

/* loop designed to display all choices for user */
for(h3=1;h3<=(p1+q1);h3++)
{
    printf("%d",h3);
    printf(".");

    /* prints string at w7[h3] on screen */
    puts(w7[h3]);

    /* stores the count of available choices in a variable b */

```

```
b=h3;
```

```

}
}

/*****
THIS PROCEDURE IS DESIGNED TO INITIALISE ARRAY W7[] TO STORE THE
OPTIONS FOR THE USER THAT ARE TO BE PROVIDED BY THE SYSTEM TO
THE USER EACH TIME AS SOON AS NEXT WORD FROM INPUT TEXT IS
TAKEN.....
*****/
```

```
chn()
```

```
{
int i2=0,j2;
```

```
/* a loop designed to initialise array w7 */
```

```
while(i2<20)
```

```
{
    j2=0;
    while(j2<20)
    {
        w7[i2][j2++]=0;
    }
    w7[i2++][j2]=0;
}
```

```

/*****
THIS PROCEDURE IS MADE TO ASK THE USER TO ENTER HIS
CHOICE TO REPLACE THE INCORRECT WORD , AND AS SOO AS
HE ENTERS THE CHOICE THAT WORD IS INSERTED INTO THE OUT PUT
FILE INSTED OF THE INCORRECT WORD.....
*****/
```

```
df()
```

```
{
int q2,k2;
```

```
/* this equalises k2 the count of available choices */
```

```

k2=b;
printf("IF YOU WANT TO SKIP PRESS : 0 \n");

printf("ENTER THE CHOICE: ");

/* this gets choice from key board */
scanf("%d",&q2);

/* a loop will go on executing to get choice until choice is
greater than k2 */
while(q2>k2)
{
printf("WRONG CHOICE ,PLEASE TRY AGAIN \n");
printf("ENTER THE CHOICE: ");

/* this gets choice from key board */
scanf("%d",&q2);
}

/* this loop is made to replace the incorrect word by the
said choice of user */
if((q2<=k2)&&(q2))
{
/* it sets the memory space for temp array */
memset(temp,0,20);

/* it copies string in w7[q2] into temp */
strcpy(temp,w7[q2]);

/* it copies string in temp into w3 */
strcpy(w3,temp);
printf("WORD IS REPLACED BY : ");
printf("%s",w3);
printf("\n");
printf("-----");
printf("\n");
}

```



```
printf("\n");
printf("\n");
printf("\n");
printf("\n");
printf("\n");
printf("\n");
printf("\n");
printf("\n");
printf("\n");
printf("\n");
printf(" TO START WITH ,PLEASE PRESS RETURN ..... "
```

```
/* gets one character from keyboard */
```

```
c=getchar();
printf("\n");
printf("\n");
printf("\n");
printf("ENTER THE FILE NAME FOR CHECKING THE SPELLINGS ");
```

```
/* gets file name as string of characters and equalise
it to a variable arr */
```

```
gets(arr);
while(c!='\n');
```

```
/* return to main program */
```

```
return;
}
```

```
*****
THIS PROCEDURE IS DESIGNED TO REPLACE THE ORIGINAL FILE WITH
THE TEMPORARY FILE IN WHICH ALL CHANGES ARE STORED ,WHEN
THE USER ALLOWS TO DO SO .....
*****
```

```
rep()
{
```

```
/* opens the file tr in read mode using pointer fp3 */
```

```
fp3=fopen("tr","r");
```

```
/* opens the file input text in write mode using pointer fp2*/
```

```

fp2=fopen(arr,"w");
/* loop is made to put words of tr file in input file
one by one until end of file tr is reached */
while(!feof(fp3))
{
    /* gets one character fro file tr having pointer fp3 */
    cl=fgetc(fp3);
    /* puts one character in variable cl into file input text
having pointer fp2 */
    fputc(cl,fp2);
}

/*****
THIS PROCEDURE IS DESIGNED TO APPEND IN THE DICTIONARY TO ENTER
R SOME NEW WORDS FOR THE USER .....
*****/

append()
{
    /* this loop will go on executing until user enters '.' */
do
{
    /* this fuction allocates memory space of 20 bytes to variable */
    mn=malloc(20);
    printf("ENTER THE WORD      ");

    /* gets word from keyboard to enter it into the dictionary */
    scanf("%s",mn);

    /* a conditional loop that will execute if character entered is
not '.' */
}

```

```
if(mn[0]!='.')
{
    /* a special procedure is called to enter word into dictionary*/
    app_dbf("wrд.dbf",mn);
}
}while(mn[0]!='.');
```

```

/*SORT2.C*/
/*****
THIS PROCEDURE IS DESIGNED TO SORT THE DICTIONARY ON
WORDS (HERE THE WORDS IS IN ORIGINAL ORDER ) IN THIS PROCEDURE FIRST
THE DICTIONARY IS STORED.
IN AN ARRAY AND qsort () IS APPLIED ON IT TO SORT THE
DICTIONARY , AFTER THIS THE SORTED DICTIONARY IS PUT INTO
A OUTPUT FILE sdic .....
*****/

#include<stdio.h>
#include<string.h>
#include<memory.h>
#include<search.h>
main()
{
char w[200][20],c;
FILE *fp,*fp2;
int i,j=0,t;
fp2=fopen("sdic","w");
fp=fopen("dic.txt","r");
i=0;
while((!feof(fp)) && ( i<=200))
{
j=0;
while(((c=fgetc(fp))!='\n') && (j<=19))
{
w[i][j]=c;
j++;
}
i++;
}
qsort(w,i,20,memcmp);
t=0;
for(t=0;t<i;t++)
{
fprintf(fp2,"%s\n",w[t]);
}
close(fp2);
fclose(fp);
}

```

```

/*SORT1.C*/
/*****
THIS PROCEDURE IS DESIGNED TO SORT THE DICTIONARY ON
WORDS (DICTIONARY CONTAINS THE MIRROR IMAGES OF THE WORDS) , IN THIS
PROCEDURE FIRST THE DICTIONARY IS STORED
IN AN ARRAY AND qsort () IS APPLIED ON IT TO SORT THE
DICTIONARY , AFTER THIS THE SORTED DICTIONARY IS PUT INTO
^ OUTPUT FILE sdicl .....
*****/

#include<stdio.h>
#include<string.h>
#include<memory.h>
#include<search.h>
main()
{
char w[200][20],c;
FILE *fp,*fp2;
int i,j=0,t;

/* opens the file `sdicl` in write mode using file pointer fp2*/
fp2=fopen("sdicl","w");

/* opens the file namell in read mode using file pointer fp*/
fp=fopen("namell","r");
i=0;

/* this loop is designed to store sdicl in an array w word by word*/
while((!feof(fp)) && ( i<=200))
{
j=0;

/* this loop will execute ( to store one word ) until return or
j<=19 */
while(((c=fgetc(fp))!='\n') && (j<=19))
{
w[i][j]=c;
j++;
}
i++;
}
}

```

```
/* here standard 'c' function is applied to sort words in array w*/
qsort(w,i,20,memcmp);
t=0;
/* this loop is designed to put sorted dictionary into output file
   sdic1 word by word */
for(t=0;t<i;t++)
{

    /* this function puts string in w[t] into file sdic1 (fp2) */
    fprintf(fp2,"%s\n",w[t]);
}
/* closes file sdic1 */
close(fp2);
/* closes file namel1 */
fclose(fp);
}
```

```
/*REVEL.C*/
```

```
/******  
THIS PROCEDURE IS DESIGNED TO REVERSE THE WHOLE DICTIONARY  
, IT IS BASICALLY A RECURSION PROCEDURE IN WHICH A SINGLE PROCED-  
-URE CALLS ITSELF AGAIN AND AGAIN UNTILL END OF FILE IS REACHED  
MAIN PROCEDURE OPENS dic.txt AND namell ,AND THEN CALLS rev()  
PROCEDURE , WHICH IS RECURSIVE IN NATURE .....  
******/
```

```
#include<stdio.h>  
#include<string.h>
```

```
int t=0;  
main()
```

```
{  
FILE *fp,*fpl;  
int c;
```

```
/* opens file dic.txt in read mode using pointer fp*/
```

```
fp=fopen("dic.txt","r");
```

```
/* opens file namell in write mode using pointer fpl*/
```

```
fpl=fopen("namell","w");
```

```
/* this loop calls procedure rev() again and again until t=0*/
```

```
do
```

```
{  
rev(fp,fpl);  
} while(t!=0);
```

```
/* this procedure basically a recursive in nature cause  
it calls itself and also parameters that are to be  
are the pointers of input and output files */
```

```
rev(p,p1)
```

```
{  
int c;
```

```
/* here it gets character from input file namell*/
```

```
c=fgetc(p);  
/* it is a conditional loop and executes when c is eof*/  
if (c==EOF)  
    t=1;  
/* it is a conditional loop and executes when c is eof  
   or return or blank */  
if((c!='\n') && (c!=EOF) && (c!=' '))  
/*rev calls itself*/  
rev(p,p1);  
fputc(c,p1);
```

```
/* DB1.C */
```

```
/* This program works when the user wants to append the words in the dictionary.
```

```
In this program a procedure has been written that can be called in any program and we have to compile the program containing this procedure with the program calling the procedure. */
```

```
# include <fcntl.h>  
# include <time.h>
```

```
app_dbf (dbf_name, instr)
```

```
/* these are the parameter*/  
/* file name and the string to be appended. */
```

```
char *instr, *dbf_name;
```

```
/* This structure is used to read the header of the database file. The header of the data base file is 32 bytes long containing the information like, date of last update, total number of records, records length etc. This information can be read in the structure.*/
```

```
{  
typedef struct  
{  
char magic_no;  
char last_up_yr;  
char last_mp_dt;  
char last_up_mt;  
long no_of_records;  
short header_size;  
short record_size;  
char garbage[20];  
} head;
```

```

/*      This is the structure defined in the header file
time.h */

struct tm *tim;
char ch[90], cd[31], cg, *tmpbuf;
head header;
int i, hdsiz, fa,tt;

/*      Open the data base file to which you want to append
the data. */

fa=open (dbf__nam,0__RDWR);

/*      Read the header of the data base file. */

read(fa,&header, size of(header));

/*      Allocate memory equivalent to record size. */

tmpbuf=malloc(header.record_size);

memset(tmpbuf,32,header.record_size);

/*      Copy to the memory allocated the data you want to
append. */

memcpy(&tmpbuf[0],instr,strlen(instr));

c=lseek(fa,0L,2);

/*      Write to the file the data you want to append. */

write(fa,tmpbuf,header.record_size);

/*      Update header. */

/*      Increase number of records by one. */

header.no_of_records=header.no_of_records+1L;

```

```
/* Replace time by system time. */
tt=time();
tim=localtime(&tt);
header.last_up_yr=tim->tm_year;
header.last_up_dt=tim->tm_mon;
header.last_up_mt=tim->tm_mday;
/* Seek file from the beginning. */
lseek(fa,0L,0);
/* Write updated header. */
write(fa,&header,32L);
close(fa);
}
```

Thapar Institute of Engg. & Tech.  
PATIALA-147001

CENTRAL LIBRARY

No 90549 Dt. 27-5-94