" TRANSLATOR "

(From RDL Code to Fortran Code)

A

Project report

submitted to

Department of Computer Sciences

in partial fulfillment

Of

Masters of Computer Applications

at

Thapar Institute of Engg. & Tech.

Patiala - 147001

Submitted by

BALJIT SINGH
MCA - 4/89

CMC Limited सीएमसी लिमिटेड

Jeevan Vihar, 3, Sansad Marg, New Delhi-110 001 • Phone: + 91-11-310 184 • Telex: 031-62960 • Fax: + 91-11-3746025
जीवन विहार, 3, संसद मार्ग, नई दिल्ली-110 001 • फोन: + 91-11-310 184 • टेलैक्स: 031-62960 • फैक्स: + 91-11-3746025

June 22, 1992

CERTIFICATE

This is to certify that Project Report entitled **Translator (From RDL Language to VAX-FORTRAN)** contains the work done by Mr. Baljit Singh, under the guidance of Mr. C.N. Rao. The duration of the Project was from 21.2.92 to 19.6.92 (16 weeks). This work has not been submitted to any other Institution/University for award of Degree.

G.L. SIKKA
MANAGER, IT

Program Listing

/*

PARSER CUM CODE GENERATOR

TITLE OF THE PROJECT : PARSER CUM CODE GENERATOR

Purpose : It takes RDL files as an input then parse the RDL file and then generate an equivallent FORTRAN code.

Programmed by : BALJIT SINGH

Date Written : 10-APR-1992

Calls :

Input Parameters : RDL-filename as command line argument

Remarks :

Programmer Initials :

*/

```
include "stdio.h"

include "string.h"

include "ctype.h"

include "conio.h"

include "stdlib.h"


void parse_store(FILE *fp);

int readstring(FILE *ifp1, char *string, int ty, int *tr);

void error_exi(FILE *ifp2, int i, char *err_string);

void create_table(FILE *ifp3);

struct heade *format_heads(FILE *ifp4, int head_code,
struct heade *ss);

void compute_sta(FILE *ifp5);

void function_fortran_code(FILE *ofp1, int s1, char s2,
char s3);

void print_write_st(FILE *ofp2, char wc, int cod, int j1);

void fortran_select(FILE *ofp3, int ilabel);


int isf, icf, detail_heading_stan=0, h_lines[5], cth[10],
ctf[10], p_ct[10];

int name_len=0, tit_len=0, fro_len=0, to_len=0, iwid,
ilin, pag_lim, pag_sta;

char *nam, *tit, fro[32], to[32];

struct heade {

int code;

        char *p_h;

        char *p_hf;

        struct heade *next;
```

```
                } *inf,in1;
        struct wit_str {
                        char code;
                        char fie_nam[18];
                        char fie_typ;
                        char fie_des[8];
                        struct wit_str *next;
                       } *wit,in2;
        struct sor_ctl {
                        char sor_nam[16];
                        char sor_typ;
                        int sor_ord;
                        struct sor_ctl *next;
                       } *sor,in3;
        struct com_put {
                        char *expr;
                        struct com_put *next;
                       } *comp,in4;
```

CHAPTER 1

PARSER CUM CODE GENERATOR (MAIN PROGRAM)

```
/*
TITLE OF THE ROUTINE :  MAIN

Purpose              :  This is the main program . It
                        calls the parsing routines and
                        then generate the FORTRAN code.

Programmed by        :  BALJIT SINGH

Date Written         :  25-APR-1992

Calls                :  PARSE_STORE,READSTRING,ERROR_EXI,
                        CREATE_TABLE , FORMAT_HEADS ,
                        COMPUTE_STA,FUNCTION_FORTRAN_CODE,
                        PRINT_WRITE_ST , FORTRAN_SELECT

Input Parameters     :  RDL-filename as command line
                        arguments

Remarks              :

Programmer Initials  :

*/
```

```
main(argc,argv)
int argc;
char *argv[];
{
FILE *fp,*fpo;
char wc,*re_ph,*re_phf,outpu[32]; /* cc */
int i,ii,j1,j2,j3;
struct wit_str *twit;
struct heade *ss2;
struct com_put *tcomp;
struct sor_ctl *tsor;
if (argc<2) {
     printf("\nPLEASE ENTER THE FILE NAME ALSO\n");
     exit(0);
}
if (argc>3) {
     printf("\nERROR IN COMMAND SYNTEX\n");
     exit(0);
}
if ((fp=fopen(argv[1],"r")) == NULL) {
     printf("CANNOT OPEN FILE = %s\n",argv[1]);
     exit(0);
}
clrscr();
for(i=0;i<10;i++) {
```

```
            cth[i]=0;
            ctf[i]=0;
            p_ct[i]=0;
            if (i<5)
            h_lines[i]=0;
      }
      parse_store(fp);
      /* scanf("%d",&ii);*/
      fclose(fp);
      i=0;
      if (argc==3) {
            strcpy(outpu,argv[2]);
      }
      else {
            while(argv[1][i]!='.')
                  outpu[i]=argv[1][i++];
            outpu[i++]='.';
            outpu[i++]='F';
            outpu[i++]='O';
            outpu[i++]='R';
            outpu[i]=0;
      }
      if ((fpo=fopen(outpu,"w")) == NULL) {
            printf("error in o/p file");
            exit(0);
```

```
        }
        if ((re_ph=(char *)malloc(400)) == NULL) {
             printf("error in memory allocation");
             exit(0);
        }
        if ((re_phf=(char *)malloc(400)) == NULL) {
             printf("Error in memory allocation");
             exit(0);
        }
        j1=0;
        j2=0;
        twit = wit;
        while (twit != NULL) {
        if (twit->code=='W') {
             strcpy(re_ph+j1,twit->fie_nam);
             j1=j1+strlen(twit->fie_nam);
             *(re_ph+j1++) = ',';
             *(re_phf+j2++) = twit->fie_typ;
             strcpy(re_phf+j2,twit->fie_des);
             j2=j2+strlen(twit->fie_des);
             *(re_phf+j2++) = ',';
        }
        switch (twit->fie_typ) {
             case ('C'):
                    printf("\n character");
```

```
                fprintf(fpo,"\n character");
                break;
        case ('I'):
                printf("\n integer");
                fprintf(fpo,"\n integer");
                break;
        case ('F'):
                printf("\n real");
                fprintf(fpo,"\n real");
                break;
    }
    if (twit->fie_typ!='X') {
        if (strcmp(twit->fie_des," ")!=0) {
         if (twit->fie_typ=='I') {
                printf("*4");
                fprintf(fpo,"*4");
         }
         else {
                printf("*%s",twit->fie_des);
                fprintf(fpo,"*%s",twit->fie_des);
         }
        }
        printf(" %s",twit->fie_nam);
        fprintf(fpo," %s",twit->fie_nam);
    }
```

```
        twit = twit->next;
        }
        *(re_ph+(--j1)) = 0;
        *(re_phf+(--j2)) = 0;
        if (icf>0) {
            printf("\n dimension cth(10),ctf(10)");
            fprintf(fpo,"\n dimension cth(10),ctf(10)");
            printf("\n integer*2 cth,ctf");
            fprintf(fpo,"\n integer*2 cth,ctf");
            printf("\n data cth/%d",cth[0]);
            fprintf(fpo,"\n data cth/%d",cth[0]);
            for(i=1;i<10;i++) {
             printf(",%d",cth[i]);
             fprintf(fpo,",%d",cth[i]);
            }
            printf("/");
            fprintf(fpo,"/");
            printf("\n data ctf/%d",ctf[0]);
            fprintf(fpo,"\n data ctf/%d",ctf[0]);
            for(i=1;i<10;i++) {
             printf(",%d",ctf[i]);
             fprintf(fpo,",%d",ctf[i]);
            }
            printf("/");
            fprintf(fpo,"/");
```

```
        }
        printf("\n character*9 dd");
        fprintf(fpo,"\n character*9 dd");
        if (name_len>0) {
            printf("\n character*20 nam");
            fprintf(fpo,"\n character*20 nam");
        }
        if (tit_len>0) {
            printf("\n character*52 tit");
            printf("\n tit = \'%s\'",tit);
            fprintf(fpo,"\n character*52 tit");
            fprintf(fpo,"\n tit = \'%s\'",tit);
        }
        if (name_len>0) {
            printf("\n nam = \'%s\'",nam);
            fprintf(fpo,"\n nam = \'%s\'",nam);
        }
        printf("\n iwid = %d",iwid);
        fprintf(fpo,"\n iwid = %d",iwid);
        printf("\n ilin = %d",ilin);
        fprintf(fpo,"\n ilin = %d",ilin);
        printf("\n ipag_sta = %d",pag_sta-1);
        fprintf(fpo,"\n ipag_sta = %d",pag_sta-1);
        if (pag_lim>0) {
            printf("\n ipag_lim = %d",pag_lim);
```

```
        fprintf(fpo,"\n ipag_lim = %d",pag_lim);
    }
    printf("\n ipag = ipag_sta + 1");
    fprintf(fpo,"\n ipag = ipag_sta + 1");
    printf("\n\n call date (dd)");
    fprintf(fpo,"\n\n call date (dd)");
    ii=(iwid-20-tit_len)/2;
    printf("\n\nC **** READ FORMAT **** ");
    fprintf(fpo,"\n\nC **** READ FORMAT **** ");
    printf("\n50 format(");
    fprintf(fpo,"\n50 format(");
    twit = wit;
    j3=16;
    while ((twit != NULL)&&(twit->code=='W')) {
    j3 = j3+strlen(twit->fie_des)+1;
    if (j3<71) {
        if (twit->fie_typ=='X') {
         printf("%sX",twit->fie_des);
         fprintf(fpo,"%sX",twit->fie_des);
        }
        else {
         if (twit->fie_typ=='C') {
         printf("A%s",twit->fie_des);
         fprintf(fpo,"A%s",twit->fie_des);
         }
```

T.I.E.T

TIET

```
                else {
                        printf("%c%s",twit->fie_typ,twit->fie_des);
                    fprintf(fpo,"%c%s",twit->fie_typ,twit->fie_des);
                }
               }
               twit = twit->next;
               if (twit->code=='W') {
                printf(",");
                fprintf(fpo,",");
               }
               else {
                printf(")");
                fprintf(fpo,")");
               }
               j3++;
          }
          else {
               printf("\n\t1");
               fprintf(fpo,"\n\t1");
               j3 = 10;
          }
          }
          printf("\n100     format(X,A9,%2dX,A%2d,%2dX,\'Page      :
          \',I4)", ii, tit_len,ii);
          fprintf(fpo,"\n100   format(X,A9,%2dX,A%2d,%2dX,\'Page    :
          \',I4)", ii, tit_len,ii);
```

```
printf("\n90 format(\'1\')");
fprintf(fpo,"\n90 format(\'1\')");
printf("\n85 format(X,%d(\'-\'))",iwid);
fprintf(fpo,"\n85 format(X,%d(\'-\'))",iwid);
for(i=0;i<5;i++) {
     ss2=inf;
     while ((ss2->code != i+12)&&(ss2->next != NULL))
                    ss2 = ss2->next;
     j1 = 110+i*10;
     printf("\n\nC ***** ");
     fprintf(fpo,"\n\nC ***** ");
     switch (i){
            case (0):
                     printf("PAGEHEADING");
                     fprintf(fpo,"PAGEHEADING");
                     break;
            case (1):
                     printf("PAGEFOOTING");
                     fprintf(fpo,"PAGEFOOTING");
                     break;
            case (2):
                     printf("DETAILHEADING");
                     fprintf(fpo,"DETAILHEADING");
                     break;
            case (3):
```

```
                        printf("REPORTFOOTING");
                        fprintf(fpo,"REPORTFOOTING");
                        break;
                case (4):
                        printf("DETAILS");
                        fprintf(fpo,"DETAILS");
                        break;
        }
        printf(" FORMAT ****");
        fprintf(fpo," FORMAT ****");
        while (ss2->code == i+12) {
                j2 = (strlen(ss2->p_hf)) - 1;
                printf("\n%3d format(",j1);
                fprintf(fpo,"\n%3d format(",j1++);
                j3 = 16;
                for(ii=0;ii<j2;ii++) {
                        if (j3<71)
                                j3++;
                        else {
                                printf("\n\t1");
                                fprintf(fpo,"\n\t1");
                                j3 = 10;
                        }
                        printf("%c",ss2->p_hf[ii]);
                        fprintf(fpo,"%c",ss2->p_hf[ii]);
```

```
                    }
                    printf(")");
                    fprintf(fpo,")");
                    ss2 = ss2->next;
               }
          }
          ss2=inf;
          while (ss2!=NULL) {
          if (ss2->code>30) {
               i=ss2->code;
               if (ss2->code>60)      {
                printf("\n\nC   ****    CONTROLFOOTING%1d    FORMAT
****",i-60);
               fprintf(fpo,"\n\nC  ****  CONTROLFOOTING%1d   FORMAT
****",i-60);
               j1 = 300+(i-61)*10;
               }
               else {
                printf("\n\nC   ****    CONTROLHEADING%1d    FORMAT
****",i-30);
               fprintf(fpo,"\n\nC  ****  CONTROLHEADING%1d   FORMAT
****",i-30);
               j1 = 200+(i-31)*10;
               }
               while ((ss2!=NULL)&&(ss2->code== i)) {
               j2 = (strlen(ss2->p_hf)) - 1;
               printf("\n%3d format(",j1);
```

```
                fprintf(fpo,"\n%3d format(",j1++);
                j3 = 16;
                for(ii=0;ii<j2;ii++) {
                        if (j3<71)
                         j3++;
                        else {
                         printf("\n\t1");
                         fprintf(fpo,"\n\t1");
                         j3 = 10;
                        }
                        printf("%c",ss2->p_hf[ii]);
                        fprintf(fpo,"%c",ss2->p_hf[ii]);
                }
                printf(")");
                fprintf(fpo,")");
                ss2 = ss2->next;
        }
        else
                ss2=ss2->next;
        }
        if (isf>0) {
        printf("\n call LIBsPAWN(\'sort /DUPLICATES");
        fprintf(fpo,"\n call LIBsPAWN(\'sort /DUPLICATES");
        tsor=sor;
        while((tsor->sor_typ!='C')&&(tsor!=NULL)) {
```

```
        twit = wit;
        i=0;
        ii=0;
        while((twit->code=='W')&&(twit!=NULL)) {
             if (strcmp(twit->fie_nam,tsor->sor_nam)==0) {
               printf("\n\t1   /key=(position:%d,size:%s)",   i+1,
        twit->fie_des);
             fprintf(fpo,"\n\t1 /key=(position:%d,size:%s)",   i+1,
        twit->fie_des);
             break;
             }
             i += atoi(twit->fie_des);
             twit=twit->next;
        }
        tsor = tsor->next;
        }
        printf(" %s ",fro);
        fprintf(fpo," %s ",fro);
        i=0;
        while(fro[i]!='.') {
             printf("%c",fro[i]);
             fprintf(fpo,"%c",fro[i++]);
        }
        printf(".SOR \')");
        fprintf(fpo,".SOR \')");
        }
```

```
        printf("\n\n open(6, FILE=\'");
        fprintf(fpo,"\n\n open(6, FILE=\'");
        i=0;
        if (isf>0) {
             while(fro[i]!='.') {
                    printf("%c",fro[i]);
                    fprintf(fpo,"%c",fro[i++]);
             }
             printf(".SOR");
             fprintf(fpo,".SOR");
        }
        else {
             printf("%s",fro);
             fprintf(fpo,"%s",fro);
        }
        printf("\', FORM=\'FORMATTED\', STATUS=\'OLD\')");
        fprintf(fpo,"\', FORM=\'FORMATTED\', STATUS=\'OLD\')");
        if ((strcmp(to,"TERMINAL")==0)||(strcmp(to,"PRINTER")==0))
            wc='*';
        else {
            wc='7';
            printf("\n  open(7,  FILE=\'%s\',  FORM=\'FORMATTED\',
",to);
            printf("\n\t1STATUS=\'NEW\', RECL=150) ");
            fprintf(fpo,"\n          open(7,          FILE=\'%s\',
FORM=\'FORMATTED\', ",to);
```

```
        fprintf(fpo,"\n\t1STATUS=\'NEW\', RECL=150) ");
    }
    printf("\n70 read(6,50,ERR=71,IOSTAT=KK,END=590)");
    fprintf(fpo,"\n70 read(6,50,ERR=71,IOSTAT=KK,END=590)");
    twit = wit;
    j3 =45;
    while ((twit != NULL)&&(twit->code=='W')) {
    j3 = j3+strlen(twit->fie_nam);
    if (j3<71) {
        printf("%s",twit->fie_nam);
        fprintf(fpo,"%s",twit->fie_nam);
        do {
         twit = twit->next;
        } while (strcmp(twit->fie_nam,"DUMMY")==0);
        if (twit->code=='W') {
         printf(",");
         fprintf(fpo,",");
        }
       j3++;
    }
    else {
         printf("\n\t1");
         fprintf(fpo,"\n\t1");
         j3 = 10;
    }
```

```
}
printf("\n goto 72 ");
fprintf(fpo,"\n goto 72 ");
printf("\n71 write(*,*)\'ERROR IN READING RECORD \',KK");
fprintf(fpo,"\n71  write(*,*)\'ERROR  IN  READING  RECORD
\',KK");
printf("\n goto 70 ");
fprintf(fpo,"\n goto 70 ");
printf("\n72 continue ");
fprintf(fpo,"\n72 continue ");
fortran_select(fpo,70);
function_fortran_code(fpo,1,';',' ');
ii=0;
for(i=0;i<icf;i++)
     ii +=cth[i];
ii += h_lines[0]+h_lines[1]+h_lines[2];

printf("\n ilc=%d",ii+1);
fprintf(fpo,"\n ilc=%d",ii+1);
printf("\n write(%c,90)",wc);
fprintf(fpo,"\n write(%c,90)",wc);
printf("\n write(%c,100)dd,tit,ipag",wc);
fprintf(fpo,"\n write(%c,100)dd,tit,ipag",wc);
if (h_lines[0]>0)
     print_write_st(fpo,wc,12,110);
printf("\n write(%c,85)",wc);
```

```
fprintf(fpo,"\n write(%c,85)",wc);
if (h_lines[2]>0)
     print_write_st(fpo,wc,14,130);
printf("\n write(%c,85)",wc);
fprintf(fpo,"\n write(%c,85)",wc);
for(i=0;i<icf;i++)
     if (cth[i]>0) {
          print_write_st(fpo,wc,31+i,200+i*10);
     }
twit = wit;
while (twit != NULL) {
if (twit->code == 'N') {
     printf("\n %s = %s",twit->fie_nam,twit->fie_nam+3);
          fprintf(fpo,"\n          %s          =
%s",twit->fie_nam,twit->fie_nam+3);
}
twit = twit->next;
}
printf("\n\n500 continue");
fprintf(fpo,"\n\n500 continue");
if (pag_lim > 0) {
printf("\n   if   (ipag_sta   .ge.   %d)   then   go   to
595",pag_sta+pag_lim);
fprintf(fpo,"\n  if  (ipag_sta  .ge.  %d)  then  go  to
595",pag_sta+pag_lim);
}
printf("\n ipag_sta = ipag_sta +1");
```

```
        fprintf(fpo,"\n ipag_sta = ipag_sta +1");

        printf("\n if (ipag.eq.ipag_sta) goto 510");

        fprintf(fpo,"\n if (ipag.eq.ipag_sta) goto 510");

        printf("\n write(%c,90)",wc);

        fprintf(fpo,"\n write(%c,90)",wc);

        printf("\n write(%c,100)dd,tit,ipag_sta",wc);

        fprintf(fpo,"\n write(%c,100)dd,tit,ipag_sta",wc);

        printf("\n ilc=%d",h_lines[0]+h_lines[1]+1);

        fprintf(fpo,"\n ilc=%d",h_lines[0]+h_lines[1]+1);

        if (h_lines[0]>0)

             print_write_st(fpo,wc,12,110);

        /*    printf("\n    write(%c,85)",wc);     fprintf(fpo,"\n
        write(%c,85)",wc); if (h_lines[2]>0)
        print_write_st(fpo,wc,14,130);                    printf("\n
        write(%c,85)",wc); fprintf(fpo,"\n write(%c,85)",wc); */

        if (icf > 0) {

        printf("\n if (ijump.eq.4) goto 510");

        fprintf(fpo,"\n if (ijump.eq.4) goto 510");

        printf("\n if (ijump.eq.3) goto 900");

        fprintf(fpo,"\n if (ijump.eq.3) goto 900");

        printf("\n if (ijump.eq.2) goto 800");

        fprintf(fpo,"\n if (ijump.eq.2) goto 800");

        printf("\n if (ijump.eq.1) goto 700");

        fprintf(fpo,"\n if (ijump.eq.1) goto 700");

        printf("\n690 i5 = 0");

        fprintf(fpo,"\n690 i5 = 0");

        twit = wit;
```

```
while ((twit->code!='N')&&(twit!=NULL))

     twit = twit->next;

for(i=0;i<icf;i++) {

      printf("\n  if   ((i5.eq.0).and.(%s.ne.%s))   i5   =
%d",twit->fie_nam,twit->fie_nam+3,i+1);

     fprintf(fpo,"\n if  ((i5.eq.0).and.(%s.ne.%s))  i5  =
%d",twit->fie_nam,twit->fie_nam+3,i+1);

     twit = twit->next;

}

printf("\n if (i5.eq.0) goto 510");

fprintf(fpo,"\n if (i5.eq.0) goto 510");

printf("\n700 do 704 i7=i5,%d,1",icf);

fprintf(fpo,"\n700 do 704 i7=i5,%d,1",icf);

printf("\n704 ilc = ilc + ctf(i7) ");

fprintf(fpo,"\n704 ilc = ilc + ctf(i7) ");

printf("\n if (ilc.lt.ilin) goto 710");

fprintf(fpo,"\n if (ilc.lt.ilin) goto 710");

printf("\n ijump = 1");

fprintf(fpo,"\n ijump = 1");

printf("\n goto 580");

fprintf(fpo,"\n goto 580");

for(i=icf;i>0;i--) {

     printf("\n%3d continue",710+(icf-i)*10);

     fprintf(fpo,"\n%3d continue",710+(icf-i)*10);

     if (ctf[i-1] > 0) {

     print_write_st(fpo,wc,60+i,300+(i-1)*10);
```

```
        function_fortran_code(fpo,1,'C',i+48);
        }
        twit=wit;
        while (twit!=NULL) {
        if ((twit->code=='N')&&(twit->fie_nam[1]==i+48)) {
                        printf("\n          %s          =
%s",twit->fie_nam,twit->fie_nam+3);
                       fprintf(fpo,"\n          %s          =
%s",twit->fie_nam,twit->fie_nam+3);
                break;
        }
        twit = twit->next;
        }
        printf("\n if (i5.eq.%1d) goto %d",i,800);
        fprintf(fpo,"\n if (i5.eq.%1d) goto %d",i,800);
}
printf("\n800 do 804 i7=i5,%d,1",icf);
fprintf(fpo,"\n800 do 804 i7=i5,%d,1",icf);
printf("\n804 ilc = ilc + cth(i7) ");
fprintf(fpo,"\n804 ilc = ilc + cth(i7) ");
printf("\n if (ilc.lt.ilin) goto 810");
fprintf(fpo,"\n if (ilc.lt.ilin) goto 810");
printf("\n ijump = 2");
fprintf(fpo,"\n ijump = 2");
printf("\n goto 580");
fprintf(fpo,"\n goto 580");
```

```
        for(i=icf;i>0;i--) {
             printf("\n%3d continue",810+(icf-i)*10);
             fprintf(fpo,"\n%3d continue",810+(icf-i)*10);
        /* page */
             if (p_ct[i-1] ==1) {
             printf("\n write(%c,90)",wc);
             fprintf(fpo,"\n write(%c,90)",wc);
             }
             if (ctf[i-1] > 0) {
             print_write_st(fpo,wc,30+i,200+(i-1)*10);
             function_fortran_code(fpo,1,'C',i+48);
             }
             printf("\n if (i5.eq.%1d) goto %d",i,900);
             fprintf(fpo,"\n if (i5.eq.%1d) goto %d",i,900);
        }
        }
        printf("\n900 ilc=ilc+%d",h_lines[2]);
        fprintf(fpo,"\n900 ilc=ilc+%d",h_lines[2]);
        printf("\n if (ilc .lt.  ilin) go to 910");
        fprintf(fpo,"\n if (ilc .lt.  ilin) go to 910");
        printf("\n ijump = 3");
        fprintf(fpo,"\n ijump = 3");
        printf("\n goto 580");
        fprintf(fpo,"\n goto 580");
        printf("\n910 continue");
```

```
        fprintf(fpo,"\n910 continue");
        printf("\n write(%c,85)",wc);
        fprintf(fpo,"\n write(%c,85)",wc);
        if (h_lines[2]>0)
             print_write_st(fpo,wc,14,130);
        printf("\n write(%c,85)",wc);
        fprintf(fpo,"\n write(%c,85)",wc);
        printf("\n510 ilc=ilc+%d",h_lines[4]);
        fprintf(fpo,"\n510 ilc=ilc+%d",h_lines[4]);
        printf("\n if (ilc .lt.  ilin) go to 520");
        fprintf(fpo,"\n if (ilc .lt.  ilin) go to 520");
        if (icf > 0) {
             printf("\n ijump = 3");
             fprintf(fpo,"\n ijump = 3");
        }
        printf("\n goto 580");
        fprintf(fpo,"\n goto 580");
        printf("\n520 continue");
        fprintf(fpo,"\n520 continue");
        twit=wit;
        while((twit->code!='C')&&(twit!=NULL))
             twit=twit->next;
        if (twit!=NULL) {
             tcomp=comp;
             if (tcomp->expr[0] != '=')
```

```
                tcomp = tcomp->next;
            while ((tcomp->expr[0]=='=')&&(twit->code=='C')) {
            printf("\n %s %s",twit->fie_nam,tcomp->expr);
            fprintf(fpo,"\n %s %s",twit->fie_nam,tcomp->expr);
            tcomp = tcomp->next;
            twit = twit->next;
            }
        }
        function_fortran_code(fpo,2,' ',' ');
        if (h_lines[4]>0)
            print_write_st(fpo,wc,16,150);
        else {
            printf("\n write(%c,50)",wc);
            fprintf(fpo,"\n write(%c,50)",wc);
            twit = wit;
            j3=21;
            while ((twit != NULL)&&(twit->code=='W')) {
                j3 = j3+strlen(twit->fie_nam);
                if (j3<71) {
                 printf("%s",twit->fie_nam);
                 fprintf(fpo,"%s",twit->fie_nam);
                 twit = twit->next;
                 if (twit->code == 'W') {
                        printf(",");
                        fprintf(fpo,",");
```

```
                    }
                    j3++;
                   }
                   else {
                    printf("\n\t1");
                    fprintf(fpo,"\n\t1");
                    j3=10;
                   }
          }
     }
     printf("\n75 read(6,50,ERR=76,IOSTAT=KK,END=590)");
     fprintf(fpo,"\n75 read(6,50,ERR=76,IOSTAT=KK,END=590)");
     twit = wit;
     j3 =45;
     while ((twit != NULL)&&(twit->code=='W')) {
          j3 = j3+strlen(twit->fie_nam);
          if (j3<71) {
          printf("%s",twit->fie_nam);
          fprintf(fpo,"%s",twit->fie_nam);
          do {
                    twit = twit->next;
          } while (strcmp(twit->fie_nam,"DUMMY")==0);
          if (twit->code=='W') {
                    printf(",");
                    fprintf(fpo,",");
```

```
        }
        j3++;
        }
        else {
                printf("\n\t1");
                fprintf(fpo,"\n\t1");
                j3 = 10;
        }
}
printf("\n goto 77 ");
fprintf(fpo,"\n goto 77 ");
printf("\n76 write(*,*)\'ERROR IN READING RECORD \',KK");
fprintf(fpo,"\n76  write(*,*)\'ERROR  IN  READING  RECORD
\',KK");
printf("\n goto 75 ");
fprintf(fpo,"\n goto 75 ");
printf("\n77 continue ");
fprintf(fpo,"\n77 continue ");
fortran_select(fpo,75);
if (icf>0) {
     printf("\n goto 690");
     fprintf(fpo,"\n goto 690");
}
else {
     printf("\n goto 510");
     fprintf(fpo,"\n goto 510");
```

```
        }
        printf("\n580 continue");
        fprintf(fpo,"\n580 continue");
        if (h_lines[1]>0) {
             print_write_st(fpo,wc,13,120);
             function_fortran_code(fpo,1,'P','_');
        }
        printf("\n go to 500");
        fprintf(fpo,"\n go to 500");
        printf("\n590 continue");
        fprintf(fpo,"\n590 continue");
        if (h_lines[1]>0)
             print_write_st(fpo,wc,13,120);
        printf("\n595 continue");
        fprintf(fpo,"\n595 continue");
        for(i=0;i<icf;i++)
             if (ctf[i]>0) {
                  print_write_st(fpo,wc,61+i,300+i*10);
             }
        if (h_lines[3]>0)
             print_write_st(fpo,wc,15,140);
        printf("\n END");
        fprintf(fpo,"\n END");
        fclose(fpo);
        scanf("%d",&ii);
```

```
}
```

27-5-94

CHAPTER 2

PARSER CUM CODE GENERATER (SUBPROGRAM)

```
TITLE OF THE ROUTINE  :  PARSE_STORE

Purpose               :  This parses the RDL file and
                         store the values of the data
                         objects .

Programmed by         :  BALJIT  SINGH

Date  Written         :  26-APR-1992

Calls                 :  READSTRING , ERROR_EXI ,
                         CREATE_TABLE , FORMAT_HEADS ,
                         COMPUTE_STA

Input  Parameters     :  RDL file pointer as an argument

Remarks               :

Programmer Initials   :

*/
```

Parse_store

```
void parse_store(FILE *fp)

{

char  *fstring[]  ={"REPORT",  "NAME",  "WIDTH",  "LINES",
"TITLE",  "PAGELIMIT",  "PAGESTART", "FROM", "TO", "WITH",
"SORT",    "SELECT",     "PAGEHEADING",     "PAGEFOOTING",
"DETAILHEADING",  "REPORTFOOTING",  "DETAILS", "CONTROLS",
"COMPUTE", "CONTROL", "END" };

int len,ii,i,tr1,co[22];

char *tmp,*sel;

char st[25],ch;

struct wit_str *twit,*ww;

struct sor_ctl *sor2,*tsor;

struct com_put *cc,*tcomp;


iwid = 130;

ilin = 40;

pag_sta = 1;

strcpy(fro,"datafile");

strcpy(to,"REPORT.REP");

if ((tmp=(char *)malloc(200)) == NULL) {

    printf("error in malloc 50");

    exit(0);

}

if ((nam=(char *)malloc(20)) == NULL) {

    printf("error in malloc 50");

    exit(0);

}
```

```
        if ((tit=(char *)malloc(55)) == NULL) {
            printf("error in malloc 50");
            exit(0);
        }
        for(i=0; i<21; i++)
            co[i]=0;
        do {
        len=readstring(fp,st,4,&tr1);
        for(i=0; i<22; i++)
        if (strcmp(fstring[i],st) == 0) {
        if (co[i]!=1) {
            if (i!=19)
             co[i]=1;
            puts(st);
            switch(i) {
             case 1:  /* NAME */
                    name_len=readstring(fp,nam,0,&tr1);
                    if (name_len>16)
                     error_exi(fp,1,"name");
                    len=readstring(fp,tmp,1,&tr1);
                    break;
              case 2:  /* WIDTH */
                    len=readstring(fp,tmp,3,&tr1);
                    iwid = atoi(tmp);
                    if ((iwid < 25)||(iwid > 130))
```

```
                error_exi(fp,2,"width");
               len=readstring(fp,tmp,1,&tr1);
               break;
          case 3:  /* LINES */
               len=readstring(fp,tmp,3,&tr1);
               ilin = atoi(tmp);
               if ((ilin<20)||(ilin>60))
                error_exi(fp,3,st);
               len=readstring(fp,tmp,1,&tr1);
               break;
          case 4:  /* TITLE */
               tit_len=readstring(fp,tit,2,&tr1);
               if (tit_len>51)
                error_exi(fp,4,st);
               len=readstring(fp,tmp,1,&tr1);
               break;
          case 5:  /* PAGELIMIT */
               len=readstring(fp,tmp,3,&tr1);
               pag_lim = atoi(tmp);
               if ((pag_lim<1)||(pag_lim>9999))
                error_exi(fp,5,st);
               len=readstring(fp,tmp,1,&tr1);
               break;
          case 6:  /* PAGESTART */
               len=readstring(fp,tmp,3,&tr1);
```

```
                pag_sta = atoi(tmp);
                if ((pag_sta<1)||(pag_sta>9999))
                 error_exi(fp,5,st);
                len=readstring(fp,tmp,1,&tr1);
                break;
        case 7:  /* FROM */
                fro_len=readstring(fp,fro,0,&tr1);
                if (strcmp(fro,"SORTED")==0)
                 len=readstring(fp,fro,0,&tr1);
                if (fro_len>32)
                 error_exi(fp,6,st);
                len=readstring(fp,tmp,1,&tr1);
                break;
        case 8:  /* TO */
                to_len=readstring(fp,to,0,&tr1);
                if
((strcmp(to,"PRINTER")==0)||(strcmp(to,"TERMINAL")==0))
                 strcpy(to," ");
                len=readstring(fp,tmp,1,&tr1);
                break;
        case 9:  /* WITH */
                len=readstring(fp,tmp,4,&tr1);
                if (strcmp(tmp,"FORMATS")!=0)
                 error_exi(fp,6,st);
                len=readstring(fp,tmp,4,&tr1);
                if (strcmp(tmp,"AS")!=0)
```

```
                    sor2 = sor2->next;
                   }
                   break;
                 }
                 else
                    twit = twit->next;
                 if (twit == NULL)
                    error_exi(fp,6,st);
                 len=readstring(fp,tmp,4,&tr1);
                 tsor->sor_typ=tmp[0];
                 len=readstring(fp,tmp,1,&tr1);
                } while (len>0);
                break;
        case 11:  /* nnn SELECT */
                len=readstring(fp,tmp,4,&tr1);
                if (strcmp(tmp,"IF")!=0)
                 error_exi(fp,6,st);
                if ((sel=(char *)malloc(500)) == NULL) {
                 printf("error in malloc 50");
                 exit(0);
                }
                ii=0;
                ch = getc(fp);
                do {
                 *(sel+ii++)=ch;
```

```
                    ch=getc(fp) ;
                    if (ii>500)
                            error_exi(fp,6,st);
                   } while (ch!=';');
                   *(sel+ii)=0;
                   if          ((tcomp=(struct          com_put
*)malloc(sizeof(in4))) == NULL) {
                    printf("error in malloc 50");
                    exit(0);
                   }
                   if  ((tcomp->expr=(char   *)malloc(ii+1))   ==
NULL) {
                    printf("error in malloc 40");
                    exit(0);
                   }
                   tcomp->next = NULL;
                   strcpy(tcomp->expr,sel);
                   if (comp==NULL)
                    comp=tcomp;
                   else {
                    cc=comp;
                    while (cc->next != NULL)
                            cc = cc->next;
                    cc->next = tcomp;
                   }
                   free(sel);
```

```
                break;
        case 12:  /* PAGEHEADING */
                inf = format_heads(fp,i,inf);
                break;
        case 13:  /* PAGEFOOTING */
                inf = format_heads(fp,i,inf);
                break;
        case 14:  /* DETAILHEADING */
                inf = format_heads(fp,14,inf);
                break;
        case 15:  /* REPORTFOOTING */
                inf = format_heads(fp,i,inf);
                break;
        case 16:  /* DETAILS */
                inf = format_heads(fp,i,inf);
                break;
        case 17:  /* CONTROLS ON */
                len=readstring(fp,tmp,4,&tr1);
                if (strcmp(tmp,"ON")!=0)
                 error_exi(fp,6,st);
                icf=0;
                do {
                 len=readstring(fp,tmp,4,&tr1);
                 twit = wit;
                 while (twit != NULL)
```

```
                    if (strcmp(tmp,twit->fie_nam)==0) {
                        if      ((tsor=(struct      sor_ctl
*)malloc(sizeof(in3))) == NULL) {
                         printf("error in malloc 50");
                         exit(0);
                        }
                        tsor->next = NULL;
                        strcpy(tsor->sor_nam,twit->fie_nam);
                        tsor->sor_ord = ++icf;
                        tsor->sor_typ = 'C';
                        if     ((ww      =(struct      wit_str
*)malloc(sizeof(in2))) == NULL) {
                         printf("error in malloc 7");
                         exit(0);
                        }
                        ww->next = NULL;
                        ww->code = 'N';
                        ww->fie_typ = twit->fie_typ;
                        ww->fie_nam[0] = 'C';
                        itoa(icf,ww->fie_nam+1,10);
                        ww->fie_nam[2] = '_';
                        strcpy(ww->fie_nam+3,twit->fie_nam);
                        strcpy(ww->fie_des,twit->fie_des);
                        if (sor == NULL) {
                         sor = tsor;
                         sor2 = tsor;
```

```
                    }
                    else {
                     sor2->next = tsor;
                     sor2 = sor2->next;
                    }
                    twit = wit;
                    while (twit->next != NULL)
                     twit = twit->next;
                    twit->next=ww;
                    break;
                }
                else
                    twit = twit->next;
                if (twit == NULL)
                     error_exi(fp,6,st);
                len=readstring(fp,tmp,1,&tr1);
               } while (len>0);
               break;
          case 18:  /* nnn COMPUTE */
               compute_sta(fp);
               len=readstring(fp,tmp,1,&tr1);
               break;
          case 19:  /* CONTROL HEADING */
               len=readstring(fp,tmp,4,&tr1);
               if (strcmp(tmp,"HEADING")==0)
```

```
                    ii=30;
                   else
                    if (strcmp(tmp,"FOOTING")==0)
                    ii=60;
                    else
                    error_exi(fp,6,st);
                   len=readstring(fp,tmp,4,&tr1);
                   if (strcmp(tmp,"ON")!=0)
                    error_exi(fp,6,st);
                   len=readstring(fp,tmp,4,&tr1);
                   tsor = sor;
                   while ( tsor != NULL )
                                                              if
((strcmp(tmp,tsor->sor_nam)==0)&&(tsor->sor_typ=='C'))
                           break;
                    else
                           tsor = tsor->next;
                   if (tsor == NULL)
                    error_exi(fp,6,st);
                   inf = format_heads(fp,ii+tsor->sor_ord,inf);
                   break;
             case 20:  /* END */
                   break;
          }
          break;
     }
```

```
        else
            error_exi(fp,6,st);
        }
        else
        if (i==21) error_exi(fp,6,st);
        }                    while                    ((strcmp(st,
        "END")!=0)&&(strcmp(fstring[i],"END")!=0));
        free(tmp);
        return;
        }
```

```
/*

TITLE OF THE ROUTINE  :  READSTRING

Purpose               :  This read the RDL file character
                         by character and return the total
                         number of characters  read  from
                         the file.

Programmed by         :  BALJIT  SINGH

Date  Written         :  05-MAY-1992

Calls                 :  ERROR_EXI

Input  Parameters     :  RDL  file  pointer , pointer  of
                         string, code of string type

Remarks               :

Programmer Initials   :

*/
```

```
int readstring(FILE *ifp1, char *string,int ty,int *tr)
{
int i;
static int ch;

while ((isspace(ch)!=0)&&(ty!=1))
    ch=getc(ifp1);
i=0;
if (ch==';') {
ch=' ';
return(-1);
}
if (ty==0)
if ((ch==39)||(ch==34))
     ty=2;
else
     ty=4;
switch (ty) {
case 1:
     do {
     if (((isspace(ch)!=0)||(ispunct(ch)  !=  0))&&(ch  !=
EOF)&&(ch != 39)&&(ch!=34))
     *(string+i++)=ch;
     ch=getc(ifp1);
     if (ch==';') {
           ch=' ';
```

```
                return(-1);
            }
             } while (((isspace(ch) != 0)||(ispunct(ch) != 
        0))&&(ch != EOF)&&(ch != 39)&&(ch!=34));
            break;
        case 2:
            if ((ch==39)||(ch==34)) {
            do {
                    if ((ch!=10)&&(ch!=EOF)&&(ch!=39)&&(ch!=34))
                     *(string+i++)=ch;
                    ch=getc(ifp1);
            } while ((ch!=39)&&(ch!=34)&&(ch!=EOF));
            ch=' ';
            }
            else
            error_exi(ifp1,6,string);
            break;
        case 3:
            do {
            if (((isdigit(ch)!=0)||(ch=='.'))&&(ch!=EOF))
                     *(string+i++)=ch;
            ch=getc(ifp1);
            } while (((isdigit(ch)!=0)||(ch=='.'))&&(ch!=EOF));
            break;
        case 4:
            do {
```

```
        if (((isalnum(ch)!=0)||(ch=='_'))&&(ch!=EOF))
                *(string+i++)=ch;
        ch=getc(ifp1);
        } while (((isalnum(ch)!=0)||(ch=='_'))&&(ch!=EOF));
        break;
    case 5:
        if (ch==24) {
        do {
                if ((ch!=25)&&(ch!=10)&&(ch!=EOF))
                *(string+i++)=ch;
                ch=getc(ifp1);
        } while ((ch!=25)&&(ch!=10)&&(ch!=EOF));
        }
        else
        error_exi(ifp1,6,string);
        break;
    case 6:
        do {
        if ((isalpha(ch)!=0)&&(ch!=EOF))
                *(string+i++)=ch;
        ch=getc(ifp1);
        } while ((isalpha(ch)!=0)&&(ch!=EOF));
        break;
    }
    *(string+i)=0;
```

```
        string = strupr(string);

        *tr = ty;

        return(i);

        }
```

```
/*

TITLE OF THE ROUTINE  :  ERROR_EXI

Purpose               :  This prints an error message  and
                         then quit to prompt.

Programmed by         :  BALJIT  SINGH

Date  Written         :  05-MAY-1992

Calls                 :

Input  Parameters     :  Error code , error string

Remarks               :

Programmer Initials   :

*/
```

```
void error_exi(FILE *ifp2,int i,char *err_string)
{
fclose(ifp2);
printf("ERROR IN %s ::",err_string);
switch(i) {
case 1:
     printf("MAX.  LENGTH IS 16 CHARACTERS");
     break;
case 2:
     printf("RANGE LIES BETWEEN 25 AND 130");
     break;
case 3:
     printf("RANGE LIES BETWEEN 20 AND 60 ");
     break;
case 4:
     printf("MAX.  LENGTH IS 51 CHARACTERS");
     break;
case 5:
     printf("range lies between 1 AND 9999");
     break;
case 6:
     printf("ERROR IN FILENAME");
     break;
}
exit(0);
```

```
Error_exi

            return;

            }
```

```
/*

        TITLE OF THE ROUTINE  :  CREATE_TABLE

        Purpose               :  This parses the WITH statement of
                                 RDL program and then creates  the
                                 symbol table and store  the  data
                                 objects and its information in
        the
                                 table.

        Programmed by         :  BALJIT  SINGH

        Date  Written         :  05-MAY-1992

        Calls                 :  ERROR_EXI , READSTRING

        Input  Parameters     :  RDL file pointer as an argument

        Remarks               :

        Programmer  Initials  :

        */
```

```
void create_table(FILE *ifp3)
{
char *tmp,*tp;
int len,tr;
struct wit_str *twit,*ww;
if ((tmp=(char *)malloc(100)) == NULL) {
    printf("error in malloc 4");
    exit(1);
}
if ((tp=(char *)malloc(100)) == NULL) {
    printf("error in malloc 3");
    exit(1);
}
do {
if ((twit =(struct wit_str *)malloc(sizeof(in2))) == NULL)
{
     printf("error in malloc 7");
     exit(0);
}
twit->next = NULL;
len=readstring(ifp3,tp,4,&tr);
twit->code = 'W';
if (len>0)
     if (strcmp(tp,"DUMMY")==0) {
     strcpy(twit->fie_nam,tp);
     len = readstring(ifp3,tmp,3,&tr);
```

```
                twit->fie_typ='X';
                strcpy(twit->fie_des,tmp);
                }
                else {
                ww = wit;
                while (ww != NULL)
                if (strcmp(ww->fie_nam,tp)== 0)
                        error_exi(ifp3,6,"DUPLICATE FIELDS");
                else
                        ww=ww->next;
                len = readstring(ifp3,tmp,4,&tr);
                if (tmp[0] == 'R')
                        tmp[0] = 'F';
                twit->fie_typ=tmp[0];
                strcpy(twit->fie_nam,tp);
                len = readstring(ifp3,tmp,3,&tr);
                strcpy(twit->fie_des,tmp);
                }
        if (wit == NULL) {
                wit = twit;
        }
        else {
                ww = wit;
                while (ww->next != NULL)
                        ww = ww->next;
```

```
            ww->next = twit;
        }
        len=readstring(ifp3,tmp,1,&tr);
        } while (len>0);
        free(tmp);
        free(tp);
        return;
        }
```

```
/*

TITLE OF THE ROUTINE  :  FORMAT_HEADS

Purpose               :  This parses the PAGEHEADING ,
                         PAGEFOOTING , REPORTFOOTING ,
                         DETAILHEADING , DETAILS ,
                         CONTROLHEADING , CONTROLFOOTING
                         statements of RDL program and
                         then convert it into an
                         equivallent fortran statement.

Programmed by         :  BALJIT  SINGH

Date  Written         :  05-MAY-1992

Calls                 :  READSTRING , ERROR_EXI

Input  Parameters     :  RDL file pointer, head code, and
                         pointer of structure heade

Remarks               :

Programmer  Initials  :

*/
```

```
struct heade *format_heads(FILE *ifp4,int head_code,struct
heade *ss)

{

char tm[5][70],tmp[100],st[25],c2[3]=" ",*phe,*phef;

char *func[] ={"LAST","AVG","MAX","MIN","SUM","COUNT"};

int ph1,i,jw,jfa,jf,len,tr1,j,lin;

struct heade *ss2,*ss4,*ss6;

struct wit_str *ww,*twit;

lin=0;

len=readstring(ifp4,tmp,1,&tr1);

if (len<0)

    return(ss);

if((phe=(char *)malloc(400)) == NULL) {

    printf("error in malloc 5");

    exit(0);

}

if((phef=(char *)malloc(400)) == NULL) {

    printf("error in malloc 10");

    exit(0);

}

ss6=NULL;

if (ss!=NULL) {

ss6=ss;

while (ss->next!=NULL)

     ss = ss->next;

ss2=ss;
```

```
        }
        ph1=0;
        jf=0;
        jw=0;
        jfa=0;
        len=readstring(ifp4,tmp,0,&tr1);
        if ((head_code==14)&&(strcmp(tmp,"STANDARD")==0)) {
        detail_heading_stan=1;
        do {
             len=readstring(ifp4,tmp,1,&tr1);
        } while(len>0);
        return(ss6);
        }
        if ((head_code>29)&&(strcmp(tmp,"PAGE")==0)) {
        p_ct[head_code-31]=1;
        len=readstring(ifp4,tmp,1,&tr1);
        if (len<0)
             return(ss6);
        else
             len=readstring(ifp4,tmp,0,&tr1);
        }
        do {
        strcpy(tm[0],tmp);
        if (strcmp(tmp,"SKIP")==0) {
             if ((jw>0)||(jfa>0)||(jf>1)) {
```

```
        if ((ss4=(struct heade *)malloc(sizeof(in1))) ==
NULL) {
                printf("error in malloc 15");
                exit(0);
        }
        lin += 1;
        ss4->next = NULL;
        if (ss6==NULL) {
                ss6=ss4;
                ss2=ss4;
        }
        else {
                ss2->next = ss4;
                ss2 = ss2->next;
        }
        ss2->code = head_code;
        if (jw>0) {
        *(phe+jw) = 0;
        if ((ss2->p_h=(char *)malloc(jw+1)) == NULL) {
                printf("error in malloc 20");
                exit(0);
        }
        strcpy(ss2->p_h,phe);
        }
        else
        ss2->p_h = NULL;
```

```
                if ((jfa>0)||(jf>1)) {
                        *(phef+jf) = 0;
                        if ((ss2->p_hf=(char *)malloc(jf+1)) ==  NULL)
        {
                         printf("error in malloc 25");
                         exit(0);
                        }
                        strcpy(ss2->p_hf,phef);
                }
                else
                        ss2->p_hf = NULL;
                jf=0;
                jfa=0;
                jw=0;
                }
                len=readstring(ifp4,tmp,3,&tr1);
                if ((atoi(tmp)<0)||(atoi(tmp)>5))
                error_exi(ifp4,6,st);
                for(i=0;i<atoi(tmp);i++)
                *(phef+jf++) = '/';
                lin += i;
                if (i>0)
                *(phef+jf++) = ',';
        }
        else {
                if (atoi(tmp)!=0) {
```

```
                i=atoi(tmp);
                ph1=3;
                len=readstring(ifp4,tm[1],2,&tr1);
                if (len*i > iwid)
                        error_exi(ifp4,6,st);
                }
                else {
                if (tr1==2) {
                        if (len>61)
                        error_exi(ifp4,6,st);
                        ph1=2;
                }
                else {
                        ph1=4;
                                                                if
        ((head_code==13)||(head_code==15)||(head_code>59)) {
                        for(i=0;i<6;i++)
                        if (strcmp(tmp,func[i])==0) {
                                len=readstring(ifp4,tm[1],4,&tr1);
                                ph1=5;
                                switch (head_code) {
                                case 13:
                                        strcpy(c2,"P");
                                        break;
                                case 15:
                                        strcpy(c2,"R");
```

```
                                        break;
                                default:
                                        c2[0]='C';
        itoa((head_code-60),(c2+1),10);
                                        break;
                                }
                                break;
                        }
                        }
                        len = readstring(ifp4,tm[4],1,&tr1);
                        twit = wit;
                        while (twit != NULL)
                        if (strcmp(tm[ph1-4],twit->fie_nam)==0) {
                        len=readstring(ifp4,tm[2],4,&tr1);
                        if (strcmp(tm[2],"FORMAT")!=0)
                                error_exi(ifp4,6,st);
                        len=readstring(ifp4,tm[2],6,&tr1);
                        len=readstring(ifp4,tm[3],3,&tr1);
                        break;
                        }
                        else
                        twit = twit->next;
                        if (twit == NULL)
                                error_exi(ifp4,6,st);
                        if (strcmp(tm[2],"C")==0)
```

```
                                strcpy(tm[2],"A");
            }
            }
            i = -1;
        }
        len=readstring(ifp4,tm[4],1,&tr1);
        if (len>0) {
            len=readstring(ifp4,tmp,0,&tr1);
            if                              ((i                              ==
-1)&&((strcmp(tmp,"COLUMN")==0)||(strcmp(tmp,"COL")==0)))
{
            i *= -1;
            len=readstring(ifp4,tmp,3,&tr1);
            j=0;
            for(i=jfa;i<atoi(tmp);i++)
            j++;
            jfa +=j;
            if (j>0) {
                    itoa(j,tmp,10);
                    strcpy((phef+jf),tmp);
                    jf = jf+strlen(tmp);
                    *(phef+jf++)='X';
                    *(phef+jf++)=',';
            }
            len=readstring(ifp4,tmp,0,&tr1);
        }
```

```
        }
        switch (ph1) {
            case 2:
             *(phef+jf++)=39;
             strcpy(phef+jf,tm[0]);
             jf=jf+strlen(tm[0]);
             jfa +=strlen(tm[0]);
             *(phef+jf++)=39;
             *(phef+jf++)=',';
             break;
            case 3:
             *(phef+jf++)=39;
             for(i=0;i<atoi(tm[0]);i++) {
                     strcpy(phef+jf,tm[1]);
                     jf=jf+strlen(tm[1]);
                     jfa +=strlen(tm[1]);
             }
             *(phef+jf++)=39;
             *(phef+jf++)=',';
             break;
            case 4:
             strcpy(phef+jf,tm[2]);
             jf = jf+strlen(tm[2]);
             strcpy(phef+jf,tm[3]);
             jfa +=atoi(tm[3]);
```

```
                *(phef+jf++)=',';
                strcpy(phe+jw,tm[0]);
                jw=jw+strlen(tm[0]);
                *(phe+jw++)=',';
                break;
               case 5:
                ww = wit;
                while (ww->next != NULL)
                ww = ww->next;
                if ((twit=(struct wit_str  *)malloc(sizeof(in2)))  ==
        NULL) {
                        printf("error in malloc 30");
                        exit(0);
                }
                twit->next = NULL;
                ww->next = twit;
                strcpy(phef+jf,tm[2]);
                jf = jf+strlen(tm[2]);
                strcpy(phef+jf,tm[3]);
                jf = jf+strlen(tm[3]);
                jfa +=atoi(tm[3]);
                *(phef+jf++)=',';
                twit->code = 'P';
                twit->fie_nam[0]=tm[0][0];
                twit->fie_nam[1]=tm[0][1];
                strcpy(twit->fie_nam+2,c2);
```

```
                twit->fie_nam[2+strlen(c2)]='_';
                *(phe+jw++)=tm[0][0];
                *(phe+jw++)=tm[0][1];
                strcpy(phe+jw,c2);
                jw=jw+strlen(c2);
                *(phe+jw++)='_';
                strcpy(phe+jw,tm[1]);
                strcpy(twit->fie_nam+3+strlen(c2),tm[1]);
                twit->fie_typ=tm[2][0];
                strcpy(twit->fie_des,tm[3]);
                jw=jw+strlen(tm[1]);
                *(phe+jw++)=',';
                break;
        }
        ph1=1;
        } while (len>0);
        if ((jw>0)||(jfa>0)||(jf>1)) {
        lin += 1;
        if ((ss4=(struct heade *)malloc(sizeof(in1))) == NULL) {
                printf("error in malloc 35");
                exit(0);
        }
        ss4->next = NULL;
        if (ss6==NULL) {
                ss6=ss4;
```

```
            ss2=ss4;
        }
        else {
            ss2->next = ss4;
            ss2 = ss2->next;
        }
        ss2->code = head_code;
        if (jw>0) {
            *(phe+jw) = 0;
            if ((ss2->p_h=(char *)malloc(jw+1)) == NULL) {
            printf("error in malloc 40");
            exit(0);
            }
            strcpy(ss2->p_h,phe);
        } else ss2->p_h = NULL;
        if ((jfa>0)||(jf>1)) {
            *(phef+jf) = 0;
            if ((ss2->p_hf=(char *)malloc(jf+1)) == NULL) {
            printf("error in malloc 50");
            exit(0);
            }
            strcpy(ss2->p_hf,phef);
        }
        else
            ss2->p_hf = NULL;
```

```
        }
        len=readstring(ifp4,tmp,1,&tr1);
        if ((head_code>10)&&(head_code<18))
            h_lines[head_code-12]=lin;
        if ((head_code>29)&&(head_code<60))
            cth[head_code-31]=lin;
        if ((head_code>59)&&(head_code<90))
            ctf[head_code-61]=lin;
        return(ss6);
        }
```

```
/*

        TITLE OF THE ROUTINE  :  COMPUTE_STA

        Purpose               :  This parses the COMPUTE statement
                                 of the RDL program and append
                                 the symbol table.

        Programmed by         :  BALJIT SINGH

        Date Written          :  10-MAY-1992

        Calls                 :  READSTRING , ERROR_EXI

        Input Parameters      :  RDL file pointer

        Remarks               :

        Programmer Initials   :


*/
```

```
void compute_sta(FILE *ifp5)
{
int i=0,j=0,k=0,br=0,ch;
char tt[100],tt1[20],ci=' ';
struct wit_str *ww,*twit;
struct com_put *cc,*tcomp;
if (comp!=NULL) {
cc = comp;
while (cc->next != NULL)
    cc = cc->next;
}
ch=getc(ifp5);
do {
switch (ch) {
case ('*') :
case ('-') :
case ('/') :
case ('+') :
            tt[j++]=ch;
            ch=getc(ifp5);
            k--;
            break;
case ('(') :
            tt[j++]=ch;
            ch=getc(ifp5);
```

```
                    br++;
                    break;
          case (')') :
                    tt[j++]=ch;
                    ch=getc(ifp5);
                    br--;
                    break;
          case ('=') :
                    j=0;
                    tt1[i]=0;
                    ww = wit;
                    if (ww==NULL)
                    error_exi(ifp5,6,"COMPUTE");
                    else
                    while (ww->next != NULL)
                              ww = ww->next;
                         if        ((twit=(struct        wit_str
*)malloc(sizeof(in2))) == NULL) {
                    printf("error in malloc 50");
                    exit(0);
                    }
                    twit->next = NULL;
                    twit->code='C';
                    strcpy(twit->fie_nam,tt1);
                    twit->fie_typ = 'I';
                    strcpy(twit->fie_des,"8");
```

```
                    ci='y';
                    tt[j++]=ch;
                    ww->next = twit;
                    ch=getc(ifp5);
                    k=0;
                    break;
          case (10) :
                    if ((br>0)||(k!=1))
                            error_exi(ifp5,6,"COMPUTE");
                    tt[j]=0;
                    br=0;
                    i=0;
                    k=0;
                    ci=' ';
                          if       ((tcomp=(struct         com_put
*)malloc(sizeof(in4))) == NULL) {
                    printf("error in malloc 50");
                    exit(0);
                    }
                     if  ((tcomp->expr=(char  *)malloc(j+1))   ==
NULL) {
                    printf("error in malloc 40");
                    exit(0);
                    }
                    tcomp->next = NULL;
                    strcpy(tcomp->expr,tt);
```

```
                        if (comp==NULL) {
                        cc=tcomp;
                        comp=tcomp;
                        }
                        else {
                        cc->next = tcomp;
                        cc = cc->next;
                        }
                        ch=getc(ifp5);
                        break;
        case (32) :
        case (9) :
                        ch=getc(ifp5);
                        break;
        default :
                        if (isalnum(ch)) {
                        k++;
                        if (isalpha(ch)) {
                        i=0;
                        do {
                                tt1[i++]=toupper(ch);
                                tt[j++]=toupper(ch);
                                ch=getc(ifp5) ;
                        } while (isalnum(ch)||(ch=='_'));
                        if (ci=='y') {
```

```
                    tt1[i]=0;
                    ww = wit;
                    while (ww != NULL) {
                    if (strcmp(ww->fie_nam,tt1)==0) {
                    ci=' ';
                    break;
                    }
                    else
                    ww = ww->next;
                    }
                    if (ci=='y')
                    error_exi(ifp5,6,"COMPUTE");
                }
                }
                if (isdigit(ch)) {
                do {
                    tt[j++]=ch;
                    ch=getc(ifp5) ;
                } while (isdigit(ch)||(ch=='.'));
                }
                }
                break;
        }
        if ((br<0)||(k<0)||(k>1))
            error_exi(ifp5,6,"COMPUTE");
```

```
}while (ch != ';');
tt[j]=0;
if ((br>0)||(k!=1))
    error_exi(ifp5,6,"COMPUTE");
if ((tcomp=(struct com_put *)malloc(sizeof(in4))) == NULL)
{
printf("error in malloc 50");
exit(0);
}
if ((tcomp->expr=(char *)malloc(j+1)) == NULL) {
printf("error in malloc 40");
exit(0);
}
tcomp->next = NULL;
strcpy(tcomp->expr,tt);
if (comp==NULL)
comp=tcomp;
else
cc->next = tcomp;
return;
}
```

```
/*

TITLE OF THE ROUTINE  :  FUNCTION_FORTRAN_CODE

Purpose               :  This writes the fortran statement
                         equivallent to RDL Standard
                         functions like sum , max , min,
                         count ,avr , last

Programmed by         :  BALJIT  SINGH

Date  Written         :  15-MAY-1992

Calls                 :

Input  Parameters     :  Pointer of o/p file , function
                         code , first two characters of
                         function.

Remarks               :

Programmer  Initials  :

*/
```

```
void function_fortran_code(FILE *ofp1,int s1,char  s2,char
s3)

{

struct wit_str *twit;

twit = wit;

while (twit != NULL) {

if (twit->code == 'P') {

if ((twit->fie_nam[0]=='S')&&(twit->fie_nam[1]=='U'))

    switch (s1) {

     case 1:

      if  ((s2==';')||((s2  ==  twit->fie_nam[2])&&(s3  ==
twit->fie_nam[3]))) {

             printf("\n %s = 0",twit->fie_nam);

             fprintf(ofp1,"\n %s = 0",twit->fie_nam);

     }

     break;

     case 2:

      printf("\n   %s   =   %s   +   %s",   twit->fie_nam,
twit->fie_nam, strchr(twit->fie_nam,'_')+1);

      fprintf(ofp1,"\n  %s  =  %s  +  %s",  twit->fie_nam,
twit->fie_nam, strchr(twit->fie_nam,'_')+1);

     break;

    }

else

     if ((twit->fie_nam[0]=='M')&&(twit->fie_nam[1]=='A'))

     switch (s1) {

     case 1:
```

```
                if ((s2==';')||((s2 == twit->fie_nam[2])&&(s3
        == twit->fie_nam[3]))) {

                  printf("\n   %s   =   %s",   twit->fie_nam,
        strchr(twit->fie_nam,'_')+1);

                fprintf(ofp1,"\n  %s  =  %s",  twit->fie_nam,
        strchr(twit->fie_nam,'_')+1);

                }

                break;

          case 2:

                 printf("\n  if  (%s  .lt.   %s)  %s  =  %s",
        twit->fie_nam, strchr(twit->fie_nam,'_')+1, twit->fie_nam,
        strchr(twit->fie_nam,'_')+1);

                fprintf(ofp1,"\n if (%s .lt.  %s) %s  =  %s",
        twit->fie_nam, strchr(twit->fie_nam,'_')+1, twit->fie_nam,
        strchr(twit->fie_nam,'_')+1);

                break;

          }

        else

            if ((twit->fie_nam[0]=='M')&&(twit->fie_nam[1]=='I'))

          switch (s1) {

          case 1:

           if  ((s2==';')||((s2  ==  twit->fie_nam[2])&&(s3  ==
        twit->fie_nam[3]))) {

                  printf("\n   %s   =   %s",   twit->fie_nam,
        strchr(twit->fie_nam,'_')+1);

                fprintf(ofp1,"\n  %s  =  %s",  twit->fie_nam,
        strchr(twit->fie_nam,'_')+1);

                }

                break;

          case 2:
```

```
                printf("\n  if  (%s  .gt.  %s) %s  =  %s",
        twit->fie_nam, strchr(twit->fie_nam,'_')+1, twit->fie_nam,
        strchr(twit->fie_nam,'_')+1);

                fprintf(ofp1,"\n if (%s .gt.  %s) %s  =  %s",
        twit->fie_nam, strchr(twit->fie_nam,'_')+1, twit->fie_nam,
        strchr(twit->fie_nam,'_')+1);

                break;

        }

        else

            if  ((twit->fie_nam[0]=='A')&&(twit->fie_nam[1]=='V'))
        {

         printf("\nC AVERAGE %s",twit->fie_nam);

        }

        else

            if ((twit->fie_nam[0]=='C')&&(twit->fie_nam[1]=='O'))

         switch (s1) {

         case 1:

                if ((s2==';')||((s2 == twit->fie_nam[2])&&(s3
        == twit->fie_nam[3]))) {

                printf("\n %s = 0",twit->fie_nam);

                fprintf(ofp1,"\n %s = 0",twit->fie_nam);

                }

                break;

         case 2:

                        printf("\n      %s      =      %s      +
        1",twit->fie_nam,twit->fie_nam);

                fprintf(ofp1,"\n %s = %s +  1",twit->fie_nam,
        twit->fie_nam);

         break;
```

```
        }
    else
        if ((twit->fie_nam[0]=='L')&&(twit->fie_nam[1]=='A'))
        switch (s1) {
        case 1:
                if ((s2==';')||((s2 == twit->fie_nam[2])&&(s3
    == twit->fie_nam[3]))) {
                  printf("\n    %s    =    %s",twit->fie_nam,
    strchr(twit->fie_nam,'_')+1);
                  fprintf(ofp1,"\n  %s  =   %s",twit->fie_nam,
    strchr(twit->fie_nam,'_')+1);
                }
                break;
        case 2:
                  printf("\n   %s   =   %s",   twit->fie_nam,
    strchr(twit->fie_nam,'_')+1);
                fprintf(ofp1,"\n  %s  =  %s",  twit->fie_nam,
    strchr(twit->fie_nam,'_')+1);
                break;
        }
    }
    twit = twit->next;
    }
    return;
    }
```

```
/*

TITLE OF THE ROUTINE  :  PRINT_WRITE_ST

Purpose               :  This prints the WRITE statements
                         of  the  fortran  language

Programmed by         :  BALJIT  SINGH

Date  Written         :  12-MAY-1992

Calls                 :

Input  Parameters     :  Pointer of o/p file , heade code,
                         and format label.

Remarks               :

Programmer  Initials  :

*/
```

```
void print_write_st(FILE *ofp2, char wc, int cod, int j1)
{
struct heade *ss2;
int j2,j3,ii;
j1--;
ss2=inf;
while ((ss2->code!=cod)&&(ss2!=NULL))
ss2=ss2->next;
while ((ss2->code==cod)&&(ss2!=NULL)) {
j1++;
if (ss2->p_h == NULL)
     j2 = 0;
else
     j2 = (strlen(ss2->p_h)) ;
printf("\n write(%c,%3d)",wc,j1);
fprintf(ofp2,"\n write(%c,%3d)",wc,j1);
if (j2>0) {
     j3 = 21;
     for(ii=0;ii<j2-1;ii++) {
     if (j3<71)
     j3++;
     else {
     printf("\n\t1");
     fprintf(ofp2,"\n\t1");
     j3 = 10;
```

```
                }
                printf("%c",ss2->p_h[ii]);
                fprintf(ofp2,"%c",ss2->p_h[ii]);
                }
        }
        ss2 = ss2->next;
        }
        }
```

```
/*

TITLE OF THE ROUTINE  :  FORTRAN_SELECT

Purpose               :  This writes the SELECT IF
statement
                         of RDL language into the fortran
                         language.

Programmed by         :  BALJIT  SINGH

Date  Written         :  20-MAY-1992

Calls                 :

Input  Parameters     :  Pointer of o/p file, format label

Remarks               :

Programmer  Initials  :

*/
```

```
void fortran_select(FILE *ofp3,int ilabel)
{
struct com_put *tcomp;
int i,ii;
tcomp=comp;
while(tcomp!=NULL) {
if (tcomp->expr[0] != '=') {
    printf("\n if (.NOT.(");
    fprintf(ofp3,"\n if (.NOT.(");
    ii=20;
    i=0;
    while(i<=strlen(tcomp->expr)) {
     if ((tcomp->expr[i]=='<')&&(tcomp->expr[i+1]=='>')) {
     ii +=3;
     i++;
     printf(".NE.");
     fprintf(ofp3,".NE.");
     }
     else
     if ((tcomp->expr[i]=='<')&&(tcomp->expr[i+1]=='=')) {
             ii +=3;
             i++;
             printf(".LE.");
             fprintf(ofp3,".LE.");
     }
```

```
                else
                if ((tcomp->expr[i]=='>')&&(tcomp->expr[i+1]=='=')) {
                        i++;
                        ii +=3;
                        printf(".GE.");
                        fprintf(ofp3,".GE.");
                }
                else
                if (tcomp->expr[i]=='>') {
                        ii +=3;
                        printf(".GT.");
                        fprintf(ofp3,".GT.");
                }
                else
                if (tcomp->expr[i]=='<') {
                        ii +=3;
                        printf(".LT.");
                        fprintf(ofp3,".LT.");
                }
                else
                if (tcomp->expr[i]=='=') {
                        ii +=3;
                        printf(".EQ.");
                        fprintf(ofp3,".EQ.");
                }
```

```
        else

          if  ((tcomp->expr[i]==´   ´)&&(tcomp->expr[i+1]   ==
´A´)&&(tcomp->expr[i+2]  ==  ´N´)&&(tcomp->expr[i+3]== ´D´
)&&(tcomp->expr[i+4]==´ ´)) {

                i +=4;

                ii+=6;

                printf(" .AND.  ");

                fprintf(ofp3," .AND.  ");

        }

        else

          if  ((tcomp->expr[i]==´   ´)&&(tcomp->expr[i+1]   ==
´O´)&&(tcomp->expr[i+2]  ==  ´R´)&&(tcomp->expr[i+3]  == ´
´)) {

                i += 3;

                ii+=5;

                printf(" .OR.  ");

                fprintf(ofp3," .OR.  ");

        }

        else

        if  ((tcomp->expr[i]==´  ´)&&(tcomp->expr[i+1]  ==  ´
´)&&(tcomp->expr[i+2]  ==  ´  ´)&&(tcomp->expr[i+3]  ==  ´
´)&&(tcomp->expr[i+4]==´ ´)) {

                i += 3;

                ii--;

        }

        else {

               if ((tcomp->expr[i]==10)||(tcomp->expr[i]==9))
{

                printf(" ");
```

```
                    fprintf(ofp3," ");
                   }
            else {
                    printf("%c",tcomp->expr[i]);
                    fprintf(ofp3,"%c",tcomp->expr[i]);
                   }
             }
             i++;
             ii++;
             if (ii > 65) {
             ii=10;
             printf("\n\t1");
             fprintf(ofp3,"\n\t1");
             }
           }
          printf(")) goto %d",ilabel);
          fprintf(ofp3,")) goto %d",ilabel);
          break;
       }

       tcomp = tcomp->next;
       }
       return;
       }
```