MACRO

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

struct MNT  //macro name table
{
 char mname[20];
 int pp,kp,mdtp,kpdtp,pntp;
}mnt[5];

char pnt[15][15]; //all parameters name table
char apt[15][15]; //actual parameter table

struct KPDT      //keyword parameter default table
{
 char pname[15],def[15];
}kpdt[15];

struct MDT      // macro definition table
{
 char opcode[15],value[35];
}mdt[30];


int mnt_ptr,pnt_ptr,kpdt_ptr,mdt_ptr,apt_ptr;
int i,j,m,n,k=0;
char fname[20];
char buffer[80], tok1[35], tok2[35], tok3[35];
char temp[40], temp1[40], temp2[40], temp3[40];

FILE *fp;

int searchPNT(char *s)
{
 for(m=0;m<pnt_ptr;m++)
    if(strcmp(pnt[m],s)==0)
       return(m);

 return(-1);
}
```



```
int searchKPDT(char *s)
{
 for(m=0;m<kpdt_ptr;m++)
    if(strcmp(kpdt[m].pname,s)==0)
       return(m);

 return(-1);
}

int searchMNT(char *s)
{
 for(m=0;m<mnt_ptr;m++)
    if(strcmp(s,mnt[m].mname)==0)
       return(m);

 return(-1);
}

void displayPNT()
{
 printf("\n---PNT TABLE---");
 printf("\n#\tPName");
 printf("\n-----------------------------------");
 for(m=0;m<pnt_ptr;m++)
     printf("\n%d    %s",m,pnt[m]);
 printf("\n-----------------------------------");
 getch();
}

void displayMNT()
{
 printf("\n----------MACRO NAME TABLE-------------------------");
 printf("\n#\tMName\t#PP\t#KP\t#MDTP\tKPDTP\tPNTP");
 printf("\n-----------------------------------------------------");
 for(m=0;m<mnt_ptr;m++)
    printf("\n%d\t%s\t%d\t%d\t%d\t%d\t%d",
    m,mnt[m].mname,mnt[m].pp,mnt[m].kp,mnt[m].mdtp,mnt[m].kpdtp,mnt[m].pntp);
 printf("\n-----------------------------------------------------");
 getch();
}
```

```
void displayKPT()
{
 printf("\n---KEYWORD PARAMETER DEFAULT TABLE---");
 printf("\n#\tPName\tDef");
 printf("\n-------------------------------------");
 for(m=0;m<kpdt_ptr;m++)
     printf("\n%d\t%s\t%s",m,kpdt[m].pname,kpdt[m].def);
 printf("\n-------------------------------------");
 getch();
}

void printMDT()
{
 printf("\n-----------MACRO DEFINITION TABLE-------------");
 printf("\n#\tOpcode\tOperand");
 printf("\n------------------------------------------------");
 for(m=0;m<mdt_ptr;m++)
     printf("\n%d\t%s\t%s",m,mdt[m].opcode,mdt[m].value);
 printf("\n------------------------------------------------");
 getch();
}

void printAPT()
{
 printf("\n Actual Parameter name table");
 printf("\n------------------------");
 for(m=0;m<pnt_ptr;m++)
     printf("\n %s",apt[m]);
 printf("\n");
 printf("----------------------\n");
 getch();
}
```



```
void makeMDT()
{
 m=0;
 if(tok1[0]=='&')
   {
    while(m<strlen(tok1))
            {
             tok1[m]=tok1[m+1];  //left shift to remove &
             m++;
            }
   }
 k=searchPNT(tok1);
 if(k==-1)
    sprintf(temp3,"%s",tok1);
 else
    sprintf(temp3,"(P,%d)",k+1);
 m=0;
 while(m<strlen(tok2))
      {
       tok2[m]=tok2[m+1];  //left shift to remove &
       m++;
      }
 m=0;
 while(m<strlen(tok3))
      {
       tok3[m]=tok3[m+1]; //left shift to remove &
       m++;
      }
 k=searchPNT(tok2);
 if(k==-1)
   {
    printf("\nError: Parameter %s not found",tok2);   exit(0);
   }
 sprintf(temp,"(P,%d)",k+1);
 k=searchPNT(tok3);
 if(k==-1)
   {
    printf("\nError: Parameter %s not found",tok3);   exit(0);
   }
 sprintf(temp1,"%s, (P,%d)",temp,k+1);
 strcpy(mdt[mdt_ptr].opcode,temp3);
 strcpy(mdt[mdt_ptr++].value,temp1);
}
```



```
void makeAPT(int n)
{
 i=j=0;
 apt_ptr=mnt[n].pntp;

 strcat(tok2,",");
 while(tok2[j] && tok2[j]!='=')
      {
       if(tok2[j]==',')
            {
             temp[i]='\0';
             i=0;
             strcpy(apt[apt_ptr],temp);
             apt_ptr++;
            }
       else
             temp[i++] = tok2[j];
             j++;
      }

while(tok2[j])
      {
       if(tok2[j]=='=')
            {
             temp[i]='\0';  //end of temp
             i=0; //location of temp
             apt_ptr=searchPNT(temp);
            }
       else if(tok2[j]==',')
                 {
                  temp[i]='\0';
                  i=0;
                  strcpy(apt[apt_ptr++],temp);
                  }
              else
                  temp[i++]=tok2[j];
             j++;
      }
// printAPT();
}
```



```
void expand(int n)
{
 int a,b,c,MEC,x,p;
 char t[20];
 MEC=mnt[n].mdtp;

 while(strcmp(mdt[MEC].opcode,"MEND")!=0)
      {
       strcpy(temp3,mdt[MEC].opcode);
       strcpy(tok3,temp3);
       if(tok3[0]=='(')
            {
             tok3[strlen(tok3)-1]='\0';
             c=atoi(strstr(tok3,",")+1);
             if(strcmp(apt[c-1],"")==0)
                 {
                  strcpy(t,pnt[c-1]);
                  x=searchKPDT(t);
                  strcpy(temp3,kpdt[x].def);
                 }
             else
                  sprintf(temp3,apt[c-1]);
      }

       sscanf(mdt[MEC].value,"%s %s",tok1,tok2);
       tok2[strlen(tok2)-1]='\0';
       a=atoi(strstr(tok2,",")+1);
       tok1[strlen(tok1)-2]='\0';
       b=atoi(strstr(tok1,",")+1);

       if(strcmp(apt[b-1],"")==0)
            {
             strcpy(t,pnt[b-1]);
             x=searchKPDT(t);
             sprintf(temp,"%s %s",kpdt[x].def,apt[a-1]);
            }
       else
             sprintf(temp,"%s %s",apt[b-1],apt[a-1]);

       printf("%s\t%s\n",temp3,temp);       getch();
       MEC++;
      }
}
```



```
void makeKPDT_PNT(char *s)
{
 int i=0,j=0,k=0;

 strcat(s,",");

 while(*s && *s!='=')  //for all positional parameter
      {
       if(*s==',')
          {
           temp[i]='\0';  //end of temp
           j++;  //count number of positional parameter
           i=0;  //location of temp
           k=searchPNT(temp);
           if(k==-1)
              strcpy(pnt[pnt_ptr++],temp);
           else
              {
               printf("\nError: Multiple Declaration of Symbol %s in Arg List",temp);
               exit(0);
              }
          }
        else if(*s!='&')  //donot copy &
                temp[i++]=*s;
        s++;
      }
 mnt[mnt_ptr].pp = j;
```



```
j=0;  //initialize counter again now it will count keyword parameter

while(*s)  //for all keyword parameter
     {
      if(*s=='=')
           {
            temp[i]='\0';  //end of temp
            i=0; //location of temp
            k=searchPNT(temp);
            if(k==-1)
                {
                 strcpy(pnt[pnt_ptr++],temp);
                 strcpy(kpdt[kpdt_ptr].pname,temp);
                }
            else
                {
                 printf("\nError: Multiple Declaration of Symbol %s in Arg List",temp);
                 exit(0);
                }
           }
      else if(*s==',')
           {
            temp[i]='\0';
            j++; //count number of keyword parameter
            i=0;
            strcpy(kpdt[kpdt_ptr++].def,temp);
           }
     else if(*s!='&')     //do not copy &
           temp[i++]=*s;
      s++;
     }
 mnt[mnt_ptr].kp=j;
} //end of makeKPDT_PNT() function
```



```
void separate()
{
 while(fgets(buffer,80,fp))
      {
       n=sscanf(buffer,"%s %s %s",tok1,tok2,tok3);
       if(strcmp(tok1,"MACRO")==0 && n==1)
          {
           fgets(buffer,80,fp);
           sscanf(buffer,"%s %s",tok1,tok2);
           strcpy(mnt[mnt_ptr].mname,tok1); //copy macro name
           mnt[mnt_ptr].kpdtp=kpdt_ptr;
           mnt[mnt_ptr].mdtp = mdt_ptr;
           mnt[mnt_ptr].pntp=pnt_ptr;

           makeKPDT_PNT(tok2);
          }
       else if(strcmp(tok1,"MEND")==0 && n==1)
            {
             strcpy(mdt[mdt_ptr].opcode,"MEND");
             strcpy(mdt[mdt_ptr++].value,"");
             mnt_ptr++;
            }
        else if(tok3[0]=='&' && n==3)  //creation of macro definition table
               makeMDT();
        else
            {
             k = searchMNT(tok1);
             if(k==-1)
                printf("%s",buffer);
             else
                {
                 makeAPT(k);
                 expand(k);
                }
            }
      }
}
```



```
void main(int argc,char *argv[])
{
 clrscr();
 if(argc==2)
    strcpy(fname,argv[1]);
 else
   {
    printf("\nEnter filename:");
    scanf("%s",fname);
   }
 fp=fopen(fname,"r");
 separate();
 displayMNT();
 displayPNT();
 displayKPT();
 printMDT();
 fclose(fp);
}
```

```
MACRO
COPY &Z,&W,&REG2=BREG
MOVER &REG2 &Z
MOVEM &REG2 &W
MEND
MACRO
CHANGE &X,&Y,&REG1=AREG,&OP=ADD
MOVER &REG1 &X
&OP &REG1 &Y
MOVEM &REG1 &X
MEND
READ A
COPY A,B
CHANGE A,B,REG1=CREG
COPY A,C
CHANGE C,B,OP=SUB,REG1=DREG
PRINT A
PRINT B
PRINT C
STOP
A DS 1
B DS 1
C DS 1
END
```


