```
#ifndef __STDLIB
#include <stdlib.h>
#endif
#ifndef __STRING
#include <string.h>
#endif
#ifndef _MATH_H_
#include <math.h>
#endif

#define setvarfunc(a) _FUNK=(double (*)(char *))a
#define MAXF 15
#define AUSRET(a) { *ausstr=0; return(a); }
char ausstr[200];

char *func[]={"SIN","COS","TAN","ASIN","ACOS","ATAN","LN","LOG","SQRT","EXP","SINH","COSH",
"TANH","ABS","SIGN"};
struct
{
   char name[20];
   double (*fc)(char *);
} newfunc[21];

int _ZAFC=0,errnr=0,_NACH=0,RAD=1,_LINKS=0;
double _E,_X;
double (*_FUNK)(char *)=NULL;
char *errors[]    ={{"Klammerfehler"},
                    {"Wurzel aus negativer Zahl"},
                    {"Keine Funktion"},
                    {"Syntaxfehler"},
                    {"Division duch Null"},
                    {"Keine gltige Variable"},
                    {"Endlose Rekursion"},
                    {"Kein Vergleich"},
                    {"Falsche Verknpfung"},
                    {"':'fehlt"},
                    {"'\"' fehlt"},
                    {"Variable ist weder Wert noch Ausdruck"},
                    {"Ungltige Anzahl Parameter"},
                    {"Strukturfehler"},
                    {"Keine Variablen benutzbar"},
                    {"Ungltiger Parameter"}};
char outerrstr[80];

void _strupr(char *s)
{
   char *h;
   int a;

   h=s; a=0;
   while (*h)
   {
      if (*h==34 && !a) a=1; else if (*h==34 && a) a=0;
      if (!a) *h=toupper(*h);
      h++;
   }
}

void setmul(char *s)
{
   int z,brk;
   char *i,*n,hlp[3]={0,0,0};
   char mog[9][3]={"ZX",")(","Z(","XZ","X(",")Z",")X","ZK","KZ"};
```

```
   for (i=s; *i; i++)
   {
      strncpy(hlp,i,2L);
      strupr(hlp);
      if (*hlp=='Z') *hlp='q';
      if (*(hlp+1)=='Z') *(hlp+1)='q';
      if (*hlp=='ã' || *hlp=='~') *hlp='K';
      if (*(hlp+1)=='ã' || *(hlp+1)=='~') *(hlp+1)='K';
      if (*hlp>='0' && *hlp<='9') *hlp='Z';
      if (*(hlp+1)>='0' && *(hlp+1)<='9') *(hlp+1)='Z';
      for (z=0,brk=0; (z<9 && !brk); z++)
        if (!strcmp(mog[z],hlp))
        {
           for (n=i+strlen(i); n>i; n--) *(n+1)=*n;
           *(i+1)='*';
           brk=1;
        }
   }
}

void delspac(char *s)
{
   char *i,*i2;
   int k=0;

   for (i=s; *i; i++) if (*i=='(') k++; else if (*i==')') k--;
   if (k) errnr=1;

   k=0;
   for (i=s; *i; i++)
   {
      if (*i==34 && !k) k=1; else if (*i==34 && k) k=0;
      if (*i==32 && !k)
      {
        for (i2=i; *i2; i2++) *i2=*(i2+1);
        i--;
      }
   }
}

int kpos(char c1, char c2,char *s,int pos,int dir)
{
   int kl=0;
   char *z;

   if (dir==-1)
     for (z=s+pos; *z; z++)
     {
        if (*z=='\"')
        {
           z++;
           while (*z!='\"' && *z) z++;
           if (!*z)
           {
              errnr=11;
              return(-1);
           }
        }
        if ((*z==c1 || *z==c2) && !kl) return(z-s);
        else
        if (*z=='(' || *z=='[' || *z=='{') kl++;
        else
        if (*z==')' || *z==']' || *z=='}') kl--;
        if (kl<0) errnr=1;
```

```
    }
  else
    for (z=s+(strlen(s)-1); z>=s; z--)
    {
       if (*z=='\"')
       {
          z--;
          while (*z!='\"' && z>=s) z--;
          if (z<s+pos)
          {
             errnr=11;
             return(-1);
          }
       }
       if ((*z==c1 || *z==c2) && !kl) return(z-s);
       else
       if (*z=='(' || *z=='[' || *z=='{') kl--;
       else
       if (*z==')' || *z==']' || *z=='}') kl++;
       if (kl<0) errnr=1;
    }
  if (kl) errnr=1;
  return(-1);
}

int vorz(int n,const char *s)
{
  if (n==-1) return(0);
  if ((!n) || (s[n-1]=='(') || (s[n-1]=='^') || (s[n-1]=='*') || (s[n-1]=='/') || (s[n-1]=='e'))
    return(1);
  else
    return(0);
}

int testfcn(char *s)
{
  char *x;
  int k;

  if ((*s<'A' || *s>'Z') && (*s!='@')) return(0);

  x=s;
  while ((*x>='A' && *x<='Z') || (*x=='@')) x++;
  if (*x!='(') return(0); else x++;
  k=1;
  while (k && *x)
  {
     if (*x=='(') k++;
     if (*x==')') k--;
     if (k) x++;
  }
  if (k)
  {
     /*errnr=1;*/
     return(0);
  }
  if (!*x) return(0);
  return(1);
}

double ausdruck(char *s)
{
  int n;
  char c;
```

```
   if (*s=='#')
   {
      double ret;

      _LINKS=1;
      ret=ausdruck(s+1);
      AUSRET(ret)
   }
   if (errnr || !(*s)) AUSRET(0.0)
   if ((*s=='(' || *s=='[' || *s=='{') &&
       (s[strlen(s)-1]==')' || s[strlen(s)-1]==']' || s[strlen(s)-1]=='}'))
   {
      int k=0;
      char *q;

      for (q=s+1; q[1] && k>=0; q++)
      {
         if (*q=='(' || *q=='[' || *q=='{') k++;
         else
         if (*q==')' || *q==']' || *q=='}') k--;
      }

      if (k>=0)
      {
         *q=0; s++;
      }
   }

   if (!strcmp(s,"X")) AUSRET(_X)
   if ((!strcmp(s,"PI")) ||
       (!strcmp(s, "ã"))) AUSRET(M_PI)
   if (!strcmp(s,"E")) AUSRET(_E)

   n=kpos('?','?',s,0,1);
   if (n!=-1)
   {
      char *n1,*n2,hlp[160];
      double ret;

      strcpy(hlp,s);
      hlp[n]=0;
      n1=hlp+n+1;
      n2=strchr(n1,':');
      if (n2==NULL)
      {
         errnr=10;
         AUSRET(0.0)
      }
      *n2=0; n2++;

      if (ausdruck(hlp))
         ret=ausdruck(n1);
      else
         ret=ausdruck(n2);

      AUSRET(ret)
   }

   n=kpos('&','|',s,0,1);
   if (n!=-1)
   {
      double ret1,ret2;
```

```
      n--;
      c=s[n];
      s[n]=0;

      ret1=ausdruck(s);
      ret2=ausdruck(s+n+2);

      if (c=='&' && s[n+1]=='&')
        AUSRET(ret1 && ret2)
      else
      if (c=='|' && s[n+1]=='|')
        AUSRET(ret1 || ret2)
      else
      {
         errnr=9;
         AUSRET(0.0)
      }
   }

   n=kpos('^','~',s,0,1);
   if (n!=-1)
   {
      double vor,nach;

      n--;
      c=s[n];

      if (c=='^')
      {
         s[n]=0;
         vor=ausdruck(s); nach=ausdruck(s+n+2);
         AUSRET((vor || nach) && (!vor || !nach))
      }
      else
      if (c=='~')
      {
         s[n]=0;
         vor=ausdruck(s); nach=ausdruck(s+n+2);
         AUSRET((vor && nach) || (!vor && !nach))
      }
      else
      if (s[n+1]=='~')
      {
         errnr=9;
         AUSRET(0.0)
      }
   }

   n=kpos('<','>',s,0,1);
   if (n!=-1)
   {
      int gleich;
      double ret,ret2;
      char retstr[200];

      *retstr=0;
      c=s[n];
      s[n]=0;
      if (s[n+1]=='=')
      {
         n++; gleich=1;
      }
      else gleich=0;
```

```
      ret=ausdruck(s);
      if (*ausstr) strcpy(retstr,ausstr);
      ret2=ausdruck(s+n+1);

      if (gleich)
      {
         if (c=='<')
         {
            if (*retstr && *ausstr)
            {
               if (strcmp(retstr,ausstr)<=0) AUSRET(1)
               else AUSRET(0)
            }
            else AUSRET(ret<=ret2)
         }
         else
         if (c=='>')
         {
            if (*retstr && *ausstr)
            {
               if (strcmp(retstr,ausstr)>=0) AUSRET(1)
               else AUSRET(0)
            }
            else AUSRET(ret>=ret2)
         }
      }
      else
      {
         if (c=='<')
         {
            if (*retstr && *ausstr)
            {
               if (strcmp(retstr,ausstr)<0) AUSRET(1)
               else AUSRET(0)
            }
            else AUSRET(ret<ret2)
         }
         else
         if (c=='>')
         {
            if (*retstr && *ausstr)
            {
               if (strcmp(retstr,ausstr)>0) AUSRET(1)
               else AUSRET(0)
            }
            else AUSRET(ret>ret2)
         }
      }
   }

   n=kpos('=','#',s,0,-1);
   if (n!=-1)
   {
      double ret,ret2;
      char retstr[200];

      *retstr=0;
      c=s[n];
      s[n]=0;
      if (s[n+1]=='=')
         n++;
      else
      {
         if (c=='#')
```

```
         {
            ret=!ausdruck(s+1);
            AUSRET(ret)
         }
         errnr=8;
         AUSRET(0.0)
      }

      ret=ausdruck(s);
      if (*ausstr) strcpy(retstr,ausstr);
      ret2=ausdruck(s+n+1);
      if (c=='#')
      {
         if (*retstr && *ausstr)
         {
            if (strcmp(retstr,ausstr)!=0) AUSRET(1)
            else AUSRET(0)
         }
         else AUSRET(ret!=ret2)
      }
      else
      if (c=='=')
      {
         if (*retstr && *ausstr)
         {
            if (strcmp(retstr,ausstr)==0) AUSRET(1)
            else AUSRET(0)
         }
         else AUSRET(ret==ret2)
      }
   }

   n=kpos('+','-',s,0,1);
   if (vorz(n,s)) n=kpos('+','-',s,n+1,1);
   if (n!=-1)
   {
      double ret,ret2;
      char retstr[200];

      c=s[n];
      s[n]=0;

      ret=ausdruck(s+n+1);
      strcpy(retstr,ausstr);
      ret2=ausdruck(s);

      if (c=='+')
      {
        if (*ausstr && *retstr)
        {
           strcat(ausstr,retstr);
           return((double)strlen(ausstr));
        }
        else AUSRET(ret2 + ret)
      }
      else
      {
        AUSRET(ret2 - ret)
      }
   }
   if (*s=='-')
   {
      double ret;
```

```
      ret=(-1.0)*ausdruck(s+1);
      AUSRET(ret)
   }

   n=kpos('*','/',s,0,1);
   if (n!=-1)
   {
      double help;

      c=s[n];
      s[n]=0; _NACH=1;

      help=ausdruck(s+n+1);
      _NACH=0;
      if (c=='*')
      {
         help=ausdruck(s)*help;
         AUSRET(help)
      }
      else
      if (help==0.0)
      {
         errnr=5;
         AUSRET(0.0)
      }
      else
      {
         help=ausdruck(s)/help;
         AUSRET(help)
      }
   }

   n=kpos('$','$',s,0,1);
   if (n!=-1)
   {
      double help;

      c=s[n];
      s[n]=0; _NACH=1;
      help=ausdruck(s+n+1);
      _NACH=0;
      if (help)
      {
         double h;

         h=ausdruck(s);

         AUSRET(h-(floor(h/help)*help))
      }
      else
      {
         errnr=5;
         AUSRET(0.0)
      }
   }

   n=kpos('^','^',s,0,1);
   if (n!=-1)
   {
      double vor,nach;

      s[n]=0;
      vor=ausdruck(s);
      nach=ausdruck(s+n+1);
```

```
      AUSRET(pow(vor,nach))
   }

   if (testfcn(s))
   {
      int i;
      char h[80];

      strcpy(h,s);
      *strchr(h,'(')=0;
      for (i=0; i<MAXF; i++)
         if (!strncmp(h,func[i],strlen(h)))
         {
            double hlp;

            if ((i<3 || (i>9 && i<13)) && RAD)
               hlp=ausdruck(s+strlen(h))/180.0*M_PI;
            else
               hlp=ausdruck(s+strlen(h));

            switch (i)
            {
               case 0 : AUSRET(sin(hlp))
               case 1 : AUSRET(cos(hlp))
               case 2 : AUSRET(tan(hlp))
               case 3 : if (RAD)
                           AUSRET(asin(hlp)*180.0/M_PI)
                        else
                           AUSRET(asin(hlp))
               case 4 : if (RAD)
                           AUSRET(acos(hlp)*180.0/M_PI)
                        else
                           AUSRET(acos(hlp))
               case 5 : if (RAD)
                           AUSRET(atan(hlp)*180/M_PI)
                        else
                           AUSRET(atan(hlp))
               case 6 : if (hlp<=0)
                        {
                           errnr=16;
                           AUSRET(0.0)
                        }
                        else AUSRET(log(hlp))
               case 7 : if (hlp<=0)
                        {
                           errnr=16;
                           AUSRET(0.0)
                        }
                        else AUSRET(log10(hlp))
               case 8 : if (hlp<0.0)
                           AUSRET(-sqrt(-hlp))
                        else
                           AUSRET(sqrt(hlp))
               case 9 : AUSRET(exp(hlp))
               case 10: AUSRET(sinh(hlp))
               case 11: AUSRET(cosh(hlp))
               case 12: AUSRET(tanh(hlp))
               case 13: AUSRET(fabs(hlp))
               case 14: if (hlp<0) AUSRET(-1)
                        else
                        if (!hlp)  AUSRET(0)
                        else
                        if (hlp>0) AUSRET(1)
            }
```

```
         }

      for (i=0; i<_ZAFC; i++)
         if (!strncmp(h,newfunc[i].name,strlen(h)))
            AUSRET(newfunc[i].fc(s+strlen(h)))

      errnr=3;
      sprintf(outerrstr,"\"%s\"",h);
      AUSRET(0.0)
   }
   n=strlen(s)-1;
   if (s[n]=='%')
   {
      double ret;

      s[n]=0;
      ret=ausdruck(s);
      if (s[n-1]=='%')
      {
         s[n-1]=0;
         AUSRET(ret/1000.0)
      }
      else
      AUSRET(ret/100.0)
   }
   else
   if (s[n]=='ø')
   {
      double ret;

      s[n]=0;
      ret=ausdruck(s);
      if (!RAD) AUSRET(ret/180.0*M_PI)
      else AUSRET(ret)
   }
   else
   if (s[n]=='!')
   {
      double erg;
      int i,e;

      s[n]=0;
      e=(int)(ausdruck(s));
      erg=1.0;
      for (i=2; i<=e; i++) erg*=i;
      AUSRET(erg)
   }
   else
   if (s[n]=='T' && s[n-1]>47 && s[n-1]<58) /* T ist operator */
   {
      double erg,z;
      char *p,*p2;

      s[n]=0;
      erg=0.0; z=1.0;
      p2=p=s;
      while (p2!=NULL)
      {
         p2=strchr(p,':');
         if (p2!=NULL) *p2=0;
         erg+=(atof(p)/z);
         z*=(double)60.0;
         p=p2+1;
      }
```

```
      AUSRET(erg)
   }

   if (*s>47 && *s<58)
   {
      char *j;

      if ((j=strchr(s,','))!=NULL) *j='.';
      j=s;

      while (*j && !errnr)
      {
         if ((*j<48 || *j>57) && *j!='+' && *j!='-' && *j!='E' && *j!='.')
            errnr=4;
         j++;
      }
      if (!errnr) AUSRET(atof(s)) else AUSRET(0.0)
   }
   else
   if (*s==34)  /* 34 = '\"' */
   {
      char *e,hlp[200];

      strncpy(hlp,s+1,199);
      hlp[199]=0;
      e=strrchr(hlp,34);
      if (!e)
      {
         errnr=11;
         AUSRET(0.0)
      }
      *e=0;
      strcpy(ausstr,hlp);
      return((double)strlen(ausstr));
   }
   else
   if (_FUNK) return(_FUNK(s));
   errnr=15;
   AUSRET(0.0)
}

double fx(double x,const char *s)
{
   char fs[500];

   *ausstr=errnr=0;
   strcpy(fs,s);
   _E=exp((double)(1));
   delspac(fs);
   _strupr(fs);
   _X=x;
   if (errnr) return(0.0);
   return(ausdruck(fs));
}

double f(const char *s)
{
   char fs[200];
   double erg;

   _LINKS=0;

   strcpy(fs,s);
   *ausstr=errnr=0;
```

```
   _E=exp((double)(1.0));
   delspac(fs);
   _strupr(fs);
   if (errnr) return(0.0);
   erg=ausdruck(fs);
   return(erg);
}

void addfunc(char *name,double (*fc)(char *))
{
   if (_ZAFC==21) return;
   newfunc[_ZAFC].fc=fc;
   strcpy(newfunc[_ZAFC].name,name);
   _ZAFC++;
}

char far *errstr(int errnr)
{
   char ret[300];

   if (*outerrstr)
   {
      sprintf(ret,"%s : %s",outerrstr,errors[errnr-1]);
      *outerrstr=0;
   }
   else
      strcpy(ret,errors[errnr-1]);
   return(ret);
}SUB
```