```
using System;
using System.Runtime.InteropServices;
// =+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
[StructLayout(LayoutKind.Explicit)]
public struct bift
{
    [FieldOffset(0)]
    public unsafe byte* sp;

    [FieldOffset(0)]
    public unsafe double* f;

    public unsafe bift(byte* sp1,double* f1){
        sp=sp1;
        f=f1;
    }
}
//=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
public static class cFormula
{
    //=========================================================================================
    static cFormula()
    {
    }
    public static double x;
    public static double y;
    public const int CUANFUNC=8;

    public const int tipoDat=9;

    //===================================================================================
    public static unsafe double valor(bift i)
    {
        bift i0 = new bift();
        bift i1 = new bift();
        bift i_1 = new bift();

        i0 = i1 = i_1 = i;
        i_1.sp=i.sp-1;

        if(*i_1.sp==((byte)0))      return *i.f;
        if(*i_1.sp==((byte)1))      return x;
        if(*i_1.sp==((byte)2))      return y;
        if(*i_1.sp==((byte)16))  {
            i0.sp=i.sp+2;
            i1.sp=i.sp+*i.sp;
            return valor(i0)+valor(i1);
        }
        if(*i_1.sp==((byte)17))     {
            i0.sp=i.sp+2;
            i1.sp=i.sp+*i.sp;
            return valor(i0)-valor(i1);
        }
        if(*i_1.sp==((byte)18)){
            i0.sp=i.sp+1;
            return -valor(i0);
            }
        if(*i_1.sp==((byte)19)){
            i0.sp=i.sp+2;
            i1.sp=i.sp+*i.sp;
            return valor(i0)*valor(i1);
        }
```

```
                if(*i_1.sp==((byte)20)) {
                            i0.sp=i.sp+2;
                            i1.sp=i.sp+*i.sp;
                            return valor(i0)/valor(i1);
                }
                if(*i_1.sp==((byte)21)){
                            i0.sp=i.sp+2;
                            i1.sp=i.sp+*i.sp;
                            return  Math.Pow(valor(i0),valor(i1));
                }
                if(*i_1.sp==((byte)22)){
                            i0.sp=i.sp+1;
                            return Math.Log10(valor(i0));
                }
                if(*i_1.sp==((byte)23)) {
                            i0.sp=i.sp+1;
                            return Math.Sin(valor(i0));
                }
                if(*i_1.sp==((byte)24))     {
                            i0.sp=i.sp+1;
                            return Math.Cos(valor(i0));
                }
                if(*i_1.sp==((byte)25))     {
                            i0.sp=i.sp+1;
                            return Math.Tan(valor(i0));
                }
                if(*i_1.sp==((byte)26))     {
                            i0.sp=i.sp+1;
                            return Math.Asin(valor(i0));
                }
                if(*i_1.sp==((byte)27))     {
                            i0.sp=i.sp+1;
                            return Math.Acos(valor(i0));
                }
                if(*i_1.sp==((byte)28)){
                            i0.sp=i.sp+1;
                            return Math.Atan(valor(i0));
                }
                if(*i_1.sp==((byte)29))     {
                            i0.sp=i.sp+1;
                            return Math.Sqrt(valor(i0));
                }
                //            default:   printf("Bug!\n"); exit(1);
                return 0;
        }
        //========================================================================
        public static unsafe double evalua(byte* i, double xpar, double ypar)
{
                x=xpar; y=ypar;

                bift j=new bift();
                j.sp=i+2;
                return valor(j);
}
        //========================================================================
        public static unsafe double evaluax(byte* i,double xpar)
{
                bift j=new bift();

                j.sp=i+2;
                x=xpar;
                return valor(j);
}
```

```
    //======================================================================
    public static unsafe double evaluay(byte* i,double ypar)
{
        bift j=new bift();

        j.sp=i+2;
        y=ypar;
        return valor(j);
}

    //======================================================================
    static public bool signo(char c)
    {
    return ((c=='+')||(c=='-')||(c=='*')||(c=='/'));
    }
    //======================================================================
    static public unsafe void traducecad(char* s,byte* d)
    {
        char* sd;

        for(sd=s;*sd!=0;sd++){}
        traduce(s,sd,d+1);
    }
    //======================================================================
    public static unsafe int buscafuncMath (char* s,char* sd)
    {
        string[] func;

        func = new string[CUANFUNC];
        func [0] = "log";
        func [1] = "sin";
        func [2] = "cos";
        func [3] = "tan";
        func [4] = "asin";
        func [5] = "acos";
        func [6] = "atan";
        func [7] = "sqrt";

        string sfunc;
        string sfunc2;
        sfunc = "";

        for(char* i=s;(i<sd)&&(*i!='(');i++)
        {
            sfunc = sfunc +Char.ToString ((char)(*i));
            for (int k=0; k < CUANFUNC ; ++k)
            {
                sfunc2=func[k];
                if(String.Compare(sfunc,sfunc2,false)==0) return k;
            }
        }
        return CUANFUNC;
    }
    //==========================================================================
    public static unsafe bool sub(char *s, char *sd, byte *d, char* i)
    {
        byte temp;
        bift id=new bift();
        bift  j=new bift();

        if(traduce(s,i,d+2)) return true;
        (*(d+1))++;
        temp=*(d+*(d+1));

        if(traduce(i+1,sd,d+*(d+1)+1)) return true;

        *(d-1)=(byte)((byte)*(d+1)+(byte)1+(byte)*(d+*(d+1)));

        *(d+*(d+1))=temp;

        (*(d+1))++;

        if((*(d+2)==0) && (*(d+*(d+1))==0))
        {
            *(d-1)=(byte)tipoDat;     // float=5  double=9
            id.sp=d+1;
            j=id;
            j.sp=d+1;
          *id.f=valor(j);
            *d=(byte)0;
    }
    return false;
    }
//==========================================================================
    public static unsafe bool traduce (char* s, char* sd, byte* d)
    {
        int     terpar=1;
    int pars=0;
    int maxpar=0;
        int j=0;
        char trans;
        char* i;
        bift id=new bift();


        for(i=s;i<(sd-pars);i++)
        {
        if(*i=='(')
            {
            pars++;
            }
            else
            {
            if(terpar==1)
                {
                maxpar=pars;
            terpar=0;
                }
                if(*i==')')
                {
                    pars--;
                    if(pars<maxpar)  maxpar=pars;
                    if(pars<0) return true;
            }
        }
    }
```

```
        s=s+maxpar;
        sd=sd-maxpar;
        //========================================================================
        if(sd==(s+1))
        {
                if(*s=='x')
                {
                        *d = (byte)1;
                        *(d-1)=(byte)1;
                        return false;
                }
                if(*s=='y')
                {
                        *d=(byte)2;
                        *(d-1)=(byte)1;
                        return false;
                }
}
        //======================(paso    )=====================================
        pars=0;
for(i=sd-1;(i>=s) && ((*i!='+')||(pars!=0));i--)
        {
                if(*i=='(')             pars++;
                else if(*i==')')   pars--;
}
if(i>=s)
        {
                *d=(byte)16;

                 return sub(s,sd,d,i);
}
        pars=0;
        for(i=sd-1;(i > s) && ((*i!='-')||(pars!=0)||signo(*(i-1)));i--)
        {
                 if(*i=='(') pars++;
                else if(*i==')') pars--;
}
if(i>s)
        {
                *d=(byte)(17);
                return sub(s,sd,d,i);
}
        if(*s=='-')
        {
        if(traduce(s+1,sd,d+1)) return true;
                *(d-1)=*(d+1);
                *d=(byte)(18);
                return false;
}
        pars=0;
        for(i=sd-1;(i>=s) && ((*i!='*')||(pars!=0));i--)
        {
        if(*i=='(') pars++;
        else  if(*i==')')  pars--;
}
if(i>=s)
        {
                *d=(byte)19;
                return sub(s,sd,d,i);
}
        pars=0;
for(i=sd-1;(i>=s) && ((*i!='/')||(pars!=0));i--)
        {
            if(*i=='(') pars++;
            else  if(*i==')') pars--;
    }
            if(i>=s)
            {
                    *d=(byte)20;
                    return sub(s,sd,d,i);
    }
            pars=0;
    for(i=sd-1;(i>=s) && ((*i!='^')||(pars!=0));i--)
            {
                    if(*i=='(') pars++;
                    else if(*i==')')  pars--;
            }
    if(i>=s)
            {
                    *d=(byte)21;
                     return sub(s,sd,d,i);
    }
            if(Char.IsDigit((char)(*s))|| ((char)(*s)=='.'))
                    {
                    *d = (byte)0;
                    *(d-1)=(byte)tipoDat;                    /* float=5  double=9 */
                    trans=*sd;

                    *sd=(char)0;
                    string str= new string((char*)(s));
                    double dauxpt=Double.Parse(str);

                    id.sp=d+1;
                    *id.f =dauxpt;
                    *sd=trans;
                    return false;
            }
            // BUSCA FUNCION MATEMATICA
            for(i=s;(i<sd)&&(*i!='(');i++) {}
            if(i==sd) return true;

            j=buscafuncMath(s,sd);
    if(j>CUANFUNC)    return true;
            *i='(';
            if(traduce(i,sd,d+1))   return true;
    *(d-1)=*(d+1);
            *d=(byte)(22+j);
            return false;
    }
    //========================================================================
    public static unsafe bool LeeStringToCharArray(char* entr)
    {
            string sintro = Console.ReadLine();
            if (sintro.Length == 0)  return true;
            string sintroAux=sintro;
string sintrobyte=sintro;
            for(int i=0;i < sintro.Length;i++)
            {
                            sintroAux=sintro.Remove(i+1,sintro.Length-i-1);
                            sintrobyte=sintroAux.Remove(0,sintroAux.Length-1);;
                            entr[i]=Convert.ToChar(sintrobyte);
            }
return false;
    }
```

```
	//===========================================================================
	unsafe static void Main ()
	{
		do
		{
			//Console.Clear();
			char* entr = stackalloc char[256];
			byte* s = stackalloc byte[256];

			Console.WriteLine("INGRESE LA ECUACION  (o Pluse doble [Enter] para salir) ");
			Console.Write("f(x,y)=");
			if (LeeStringToCharArray(entr)) break;
			traducecad(entr,s);
			double dValor_x;
			Console.Write("x=");
			string sdValorx=Console.ReadLine();
			if (sdValorx.Length==0) dValor_x=0;
			else dValor_x=Double.Parse(sdValorx);
			double dValor_y;
			Console.Write("y=");
			string sdValory=Console.ReadLine();
			if (sdValory.Length==0) dValor_y=0;
			else dValor_y=Double.Parse(sdValory);

			Console.WriteLine("La funcion evaluada tiene el valor de:");
			double dfvalor=(double)cFormula.evalua(s,dValor_x,dValor_y);
			Console.WriteLine("f({0},{1})={2}",dValor_x,dValor_y,dfvalor);
			Console.WriteLine("--------------------------------------------------------");
			//Console.ReadLine();
		} while (true);

	}
}
//===========================================================================
```